

Implementing Cisco Quality of Service

Volume 2

Version 2.3

Student Guide

Text Part Number: 97-2811-01



Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

 CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flip Video, Flip Video (Design), Flipshare (Design), Flip Ultra, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Store, and Flip Gift Card are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0907R)

DISCLAIMER WARRANTY: THIS CONTENT IS BEING PROVIDED "AS IS." CISCO MAKES AND YOU RECEIVE NO WARRANTIES IN CONNECTION WITH THE CONTENT PROVIDED HEREUNDER, EXPRESS, IMPLIED, STATUTORY OR IN ANY OTHER PROVISION OF THIS CONTENT OR COMMUNICATION BETWEEN CISCO AND YOU. CISCO SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. This learning product may contain early release content, and while Cisco believes it to be accurate, it falls subject to the disclaimer above.

Table of Contents

Volume 2

<i>Congestion Management</i>	5-1
Overview	5-1
Module Objectives	5-1
Introducing Queuing	5-3
Overview	5-3
Objectives	5-3
Congestion and Queuing	5-4
Queuing Algorithms	5-7
First In First Out	5-9
Priority Queuing	5-10
Round Robin	5-11
Weighted Round Robin	5-12
Deficit Round Robin	5-14
Custom Queuing	5-19
Summary	5-21
Understanding Queuing Implementations	5-23
Overview	5-23
Objectives	5-23
Queuing Components	5-24
Hardware Queue (TxQ) Size	5-27
Congestion on Software Interfaces	5-29
Queuing Implementations in Cisco IOS Software	5-30
Summary	5-31
Configuring FIFO and WFQ	5-33
Overview	5-33
Objectives	5-33
FIFO Queuing	5-34
Weighted Fair Queuing	5-36
WFQ Classification	5-39
WFQ Insertion and Drop Policy	5-42
WFQ Scheduling	5-50
Benefits and Drawbacks of WFQ	5-51
Configuring WFQ	5-52
Monitoring WFQ	5-56
Summary	5-59
Configuring CBWFQ and LLQ	5-61
Overview	5-61
Objectives	5-61
CBWFQ and LLQ	5-62
Class-Based Weighted Fair Queuing	5-63
CBWFQ Architecture	5-64
CBWFQ Benefits	5-70
Configuring and Monitoring CBWFQ	5-71
Low-Latency Queuing	5-76
LLQ Architecture	5-78
LLQ Benefits	5-79
Configuring and Monitoring LLQ	5-82
Example: Calculating LLQ Bandwidth Required for VoIP	5-84
Summary	5-87

Configuring LAN Congestion Management **5-89**

Overview	5-89
Objectives	5-89
Queuing on Catalyst Switches	5-90
Weighted Round Robin	5-99
Configuring CoS-to-Queue Mappings for PQ on Catalyst 2950 Switches	5-101
Configuring WRR on Catalyst 2950 Switches	5-102
Weighted Tail Drop	5-104
Configuring WTD on Catalyst 2960 Switches	5-106
Shaped Round Robin	5-111
Configuring SRR on Catalyst 2960 Switches	5-116
Monitoring Queuing on Catalyst 2950 and 2960 Switches	5-122
Summary	5-128
Module Summary	5-131
References	5-132
Module Self-Check	5-133
Module Self-Check Answer Key	5-135

Congestion Avoidance **6-1**

Overview	6-1
Module Objectives	6-1

Introducing Congestion Avoidance **6-3**

Overview	6-3
Objectives	6-3
Behavior of TCP Senders and Receivers	6-4
Example: Windowing in TCP	6-5
Congestion and TCP	6-6
Managing Interface Congestion with Tail Drop	6-9
Tail Drop Limitations	6-10
Summary	6-13

Introducing RED **6-15**

Overview	6-15
Objectives	6-15
Random Early Detection	6-16
RED Profiles	6-17
RED Modes	6-19
TCP Traffic Before and After RED	6-20
Applying Congestion Avoidance	6-22
Summary	6-23

Configuring Class-Based Weighted RED **6-25**

Overview	6-25
Objectives	6-25
Weighted Random Early Detection	6-26
WRED Profiles	6-30
Configuring CBWRED	6-35
Example: CBWFQ Using IP Precedence with CBWRED	6-41
Configuring DSCP-Based CBWRED	6-43
Example: CBWRED Using DSCP with CBWFQ	6-46
Monitoring CBWRED	6-48
Summary	6-49

Configuring Explicit Congestion Notification	6-51
Overview	6-51
Objectives	6-51
Explicit Congestion Notification	6-52
ECN Field Defined	6-54
ECN and WRED	6-55
Configuring ECN-Enabled WRED	6-58
Monitoring ECN-Enabled WRED	6-59
Summary	6-62
Module Summary	6-63
References	6-64
Module Self-Check	6-65
Module Self-Check Answer Key	6-69
Traffic Policing and Shaping	7-1
Overview	7-1
Module Objectives	7-1
Understanding Traffic Policing and Shaping	7-3
Overview	7-3
Objectives	7-3
Traffic Policing and Shaping Overview	7-4
Why Use Traffic Conditioners?	7-5
Example: Traffic Policing	7-7
Example: Traffic Shaping	7-8
Policing vs. Shaping	7-9
Measuring Traffic Rates	7-10
Example: Token Bucket as a Coin Bank	7-11
Single Token Bucket Class-Based Policing	7-12
Dual Token Bucket Class-Based Policing	7-13
Dual-Rate Token Bucket Class-Based Policing	7-15
Example: Dual-Rate Token Bucket as a Coin Bank	7-17
Class-Based Traffic Shaping	7-18
Cisco IOS Traffic Policing and Shaping Mechanisms	7-19
Applying Traffic Conditioners	7-21
Summary	7-24
Configuring Class-Based Policing	7-25
Overview	7-25
Objectives	7-25
Class-Based Policing Overview	7-26
Configuring Single-Rate Class-Based Policing	7-28
Example: Single Rate, Dual Token Bucket Class-Based Policing	7-32
Example: Multiaction Class-Based Policing	7-33
Configuring Dual-Rate Class-Based Policing	7-34
Example: Dual-Rate Class-Based Policing	7-37
Configuring Percentage-Based Class-Based Policing	7-38
Example: Configuring Percentage-Based Class-Based Policing	7-39
Monitoring Class-Based Policing	7-40
Summary	7-42

Configuring Class-Based Shaping	7-43
Overview	7-43
Objectives	7-43
Class-Based Shaping Overview	7-44
Traffic Shaping Methods	7-45
Configuring Class-Based Shaping	7-46
Example: Average Rate, Peak Rate	7-47
Example: Class-Based Shaping with CBWFQ	7-49
Example: Class-Based Shaping Hierarchical Policy Maps	7-50
Monitoring Class-Based Shaping	7-51
Summary	7-53
Configuring Class-Based Shaping on Frame Relay Interfaces	7-55
Overview	7-55
Objectives	7-55
Frame Relay Refresher	7-56
Frame Relay Congestion Control	7-57
Frame Relay Congestion Adaptation	7-58
FECN-to-BECN Propagation	7-59
Configuring Frame Relay Adaptive Class-Based Shaping	7-60
Example: Class-Based Shaping with Frame Relay Adaptation	7-61
Monitoring MQC-Based Frame Relay Traffic Shaping	7-62
Example: MQC-Based Frame Relay Traffic Shaping on Multipoint Main Interface	7-63
Summary	7-66
Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation	7-67
Overview	7-67
Objectives	7-68
Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation	7-69
Benefits in Deploying Frame Relay Voice-Adaptive Traffic Shaping	7-71
Prerequisites for Deploying Frame Relay Voice-Adaptive Traffic Shaping	7-72
Supported Platforms	7-73
Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation Operation	7-74
Configuring the Frame Relay Voice-Adaptive Feature	7-75
Monitoring the Frame Relay Voice-Adaptive Feature	7-79
Summary	7-81
Module Summary	7-83
References	7-84
Module Self-Check	7-85
Module Self-Check Answer Key	7-87
Link Efficiency Mechanisms	8-1
Overview	8-1
Module Objectives	8-1
Understanding Link Efficiency Mechanisms	8-3
Overview	8-3
Objectives	8-3
Link Efficiency Mechanisms Overview	8-5
Example: Indexing Operations	8-7
L2 Payload Compression	8-8
Example: L2 Payload Compression Results	8-9
Header Compression	8-10
Example: Header Compression Results	8-11
Large Packets “Freeze Out” Voice on Slow WAN Links	8-12
Link Fragmentation and Interleaving	8-14
Applying Link Efficiency Mechanisms	8-15
Summary	8-17

Configuring Class-Based Header Compression	8-19
Overview	8-19
Objectives	8-19
Header Compression Overview	8-20
Example: RTP Header Compression	8-20
Class-Based TCP Header Compression	8-21
Example: Class-Based TCP Header Compression	8-22
Class-Based RTP Header Compression	8-24
Example: Class-Based RTP Header Compression	8-25
Configuring Class-Based Header Compression	8-27
Example: Configuring Class-Based TCP Header Compression	8-28
Example: Configuring Class-Based RTP Header Compression	8-29
Monitoring Class-Based Header Compression	8-30
Summary	8-31
Configuring Link Fragmentation and Interleaving	8-33
Overview	8-33
Objectives	8-33
Fragmentation Options	8-34
Serialization Delay and Fragment Sizing	8-35
Example: Determining the Proper Fragment Size	8-36
Configuring MLP with Interleaving	8-37
Example: MLP with Interleaving	8-40
Monitoring MLP with Interleaving	8-41
FRF.12 Frame Relay Fragmentation	8-43
Configuring FRF.12 Frame Relay Fragmentation	8-45
Example: FRF.12 Frame Relay Fragmentation	8-46
Monitoring FRF.12 Frame Relay Fragmentation	8-48
Summary	8-50
Module Summary	8-51
References	8-52
Module Self-Check	8-53
Module Self-Check Answer Key	8-56

Congestion Management

Overview

Congestion can occur in many different locations within a network and is the result of many factors, including oversubscription, insufficient packet buffers, traffic aggregation points, network transit points, and speed mismatches (such as at LAN-to-WAN links). Simply increasing link bandwidth is not adequate to solve the congestion issue, in most cases. Aggressive traffic can fill interface queues and starve more fragile flows such as voice and interactive traffic. The results can be devastating for delay-sensitive traffic types, making it difficult to meet the service-level requirements these applications require. Fortunately, there are many congestion management techniques available on Cisco IOS platforms, which provide you with an effective means to manage software queues and to allocate the required bandwidth to specific applications when congestion exists. This module examines the components of queuing systems and the different congestion management mechanisms available on Cisco IOS devices.

Module Objectives

Upon completing this module, you will be able to use Cisco quality of service (QoS) queuing mechanisms to manage network congestion. This ability includes being able to meet these objectives:

- Explain the operation of basic queuing algorithms, including FIFO, priority, and round-robin queuing
- Describe hardware and software queuing on a network device
- Configure WFQ to manage congestion
- Configure CBWFQ and LLQ to manage congestion
- Configure WRR on a Cisco Catalyst switch to manage LAN congestion

Introducing Queuing

Overview

Queuing algorithms are one of the primary ways to manage congestion in a network. Network devices handle an overflow of arriving traffic by using a queuing algorithm to sort traffic and determine a method of prioritizing the traffic onto an output link. Each queuing algorithm was designed to solve a specific network traffic problem and has a particular effect on network performance. This lesson describes several queuing algorithms.

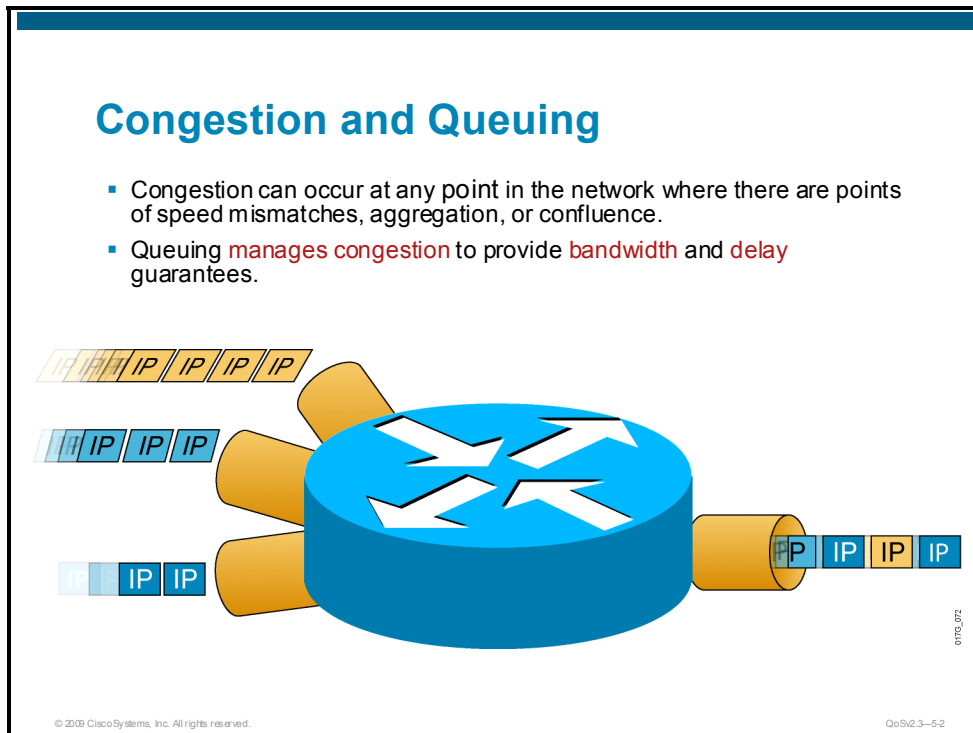
Objectives

Upon completing this lesson, you will be able to explain the operation of several queuing algorithms, including FIFO, priority, and round-robin queuing. This ability includes being able to meet these objectives:

- Explain the need for congestion management mechanisms
- List the various queuing algorithms
- Describe the FIFO queuing algorithm
- Describe the priority queuing algorithm
- Describe the round-robin queuing algorithm
- Describe the WRR queuing algorithm
- Describe the deficit round-robin queuing algorithm
- Describe the custom queuing algorithm

Congestion and Queuing

This topic describes the need for congestion management mechanisms.

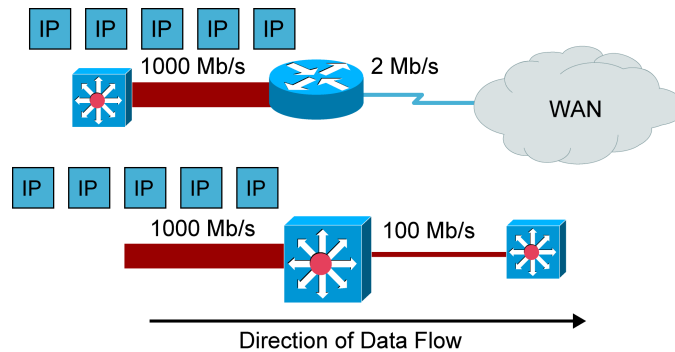


Congestion can occur anywhere within a network where there are points of speed mismatches (for example, a Gigabit Ethernet link feeding a Fast Ethernet link), aggregation (for example, multiple Gigabit Ethernet links feeding an upstream Gigabit Ethernet), or confluence (the flowing together of two or more traffic streams).

Queuing algorithms are used to manage congestion. Many algorithms have been designed to serve different needs. A well-designed queuing algorithm will provide some bandwidth and delay guarantees to priority traffic.

Congestion and Queuing: Speed Mismatch

- Speed mismatches are the **most typical cause of congestion**.
- Possibly **persistent** when going from LAN to WAN.
- Usually **transient** when going from LAN to LAN.



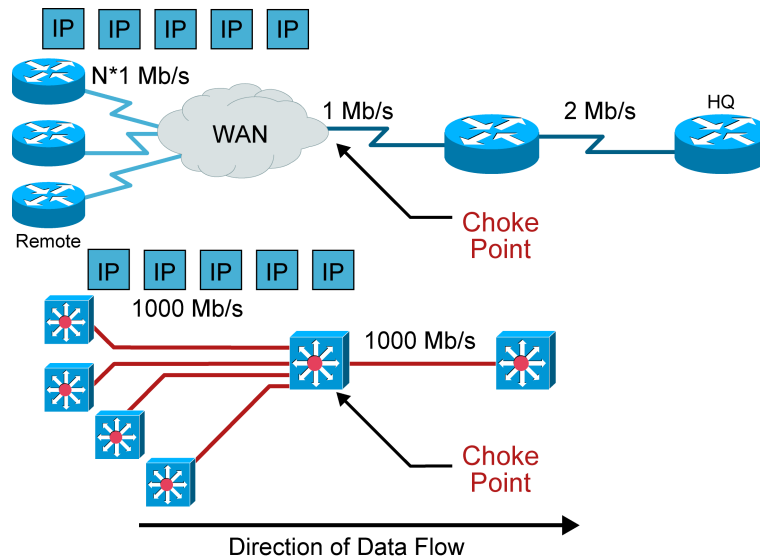
© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.3

Speed mismatches are the most typical cause of congestion in a network.

Speed mismatches are most common when traffic moves from a high-speed LAN environment (1000 Mb/s or higher) to lower-speed WAN links (1 or 2 Mb/s). Speed mismatches are also common in LAN-to-LAN environments when, for example, a 1000-Mb/s link feeds into a 100-Mb/s link.

Congestion and Queuing: Aggregation



© 2009 Cisco Systems, Inc. All rights reserved.

QoS2.3-5.4

The most common site of congestion is at points of aggregation in a network.

Typical points of aggregation occur in WANs when multiple remote sites feed back into a central services site.

In a LAN environment, congestion resulting from aggregation often occurs at the distribution layer of networks, where the different access layer devices feed traffic to the distribution-level switches.

Queuing Algorithms

This topic describes the different queuing algorithms.

Queuing Algorithms

- FIFO
- PQ
- Round robin
- WRR
- DRR
- CQ
- WFQ
- CBWFQ
- LLQ

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.5

Queuing algorithms include the following:

- **FIFO:** The simplest algorithm.
- **Priority queuing (PQ):** Allows certain traffic to be strictly prioritized.
- **Round robin:** Allows several traffic flows to share bandwidth.
- **Weighted round robin (WRR):** Allows sharing of bandwidth with preferential treatment.
- **Deficit round robin (DRR):** Resolves problem with some WRR implementations.
- **Custom queuing (CQ):** Enables you to guarantee a fixed portion of available bandwidth to specific traffic and leave the remaining bandwidth to other traffic.
- **Weighted fair queuing (WFQ):** Flow-based algorithm that simultaneously schedules interactive traffic to the front of a queue to reduce response time and fairly shares the remaining bandwidth among high-bandwidth flows.
- **Class-based weighted fair queuing (CBWFQ):** Extends WFQ functionality to provide support for user-defined traffic classes. CBWFQ allows you to specify the exact amount of bandwidth to be allocated for a specific class of traffic.
- **Low-latency queuing (LLQ):** Brings strict PQ to CBWFQ. Strict PQ allows delay-sensitive data such as voice to be dequeued and transmitted before packets in other queues are dequeued.

Queuing and IPv6

- Class-based and flow-based queuing are supported for IPv6.
- Cisco IOS Software **supports LLQ** for IPv6 QoS.
- Cisco IOS Software **does not support legacy PQ or CQ** for IPv6 QoS.
- The commands used to configure queuing for IPv6 are the same commands used to configure queuing for IPv4.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.6

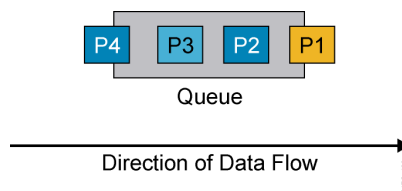
Class-based and flow-based queuing are supported for IPv6. For IPv6 QoS, Cisco IOS Software supports LLQ; however, it does not support legacy PQ or CQ. The commands used to configure queuing for IPv6 are the same commands used to configure queuing for IPv4. Refer to the *Cisco IOS Quality of Service Solutions Configuration Guide* for configuration and usage instructions for queuing features.

First In First Out

This topic describes the FIFO queuing algorithm.

First In, First Out

- First packet in is first packet out
- Simplest of all
- One queue
- All individual queues are FIFO

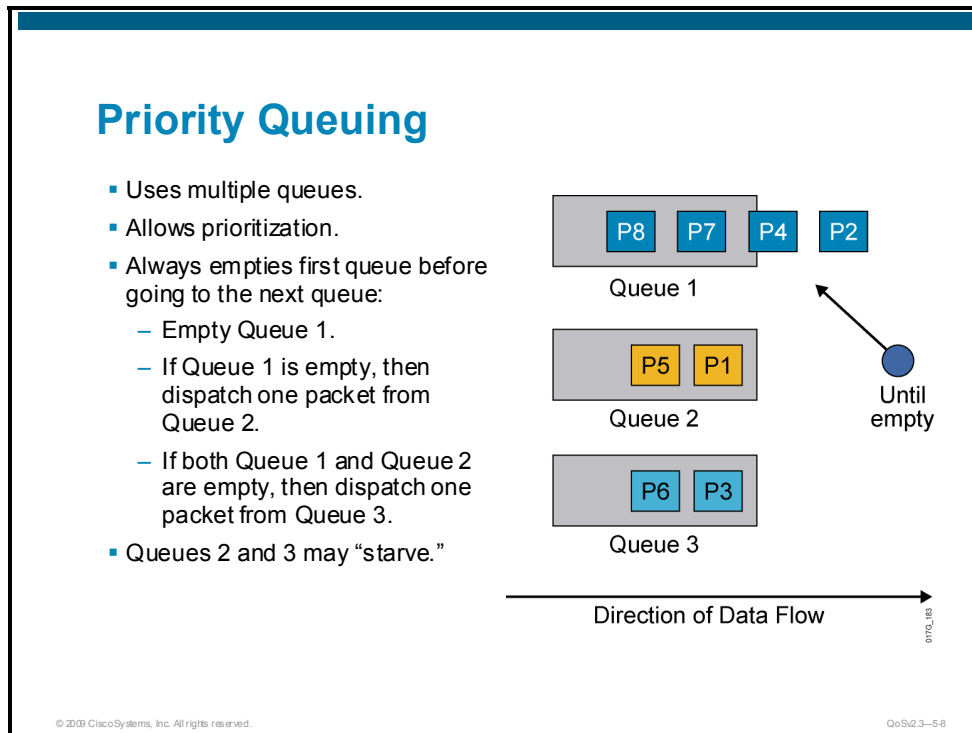


FIFO is the simplest queuing algorithm. Packets are placed into a single queue and serviced in the order they were received.

All individual queues are, in fact, FIFO queues. Other queuing methods rely upon FIFO as the congestion management mechanism for single queues, while using multiple queues to perform more advanced functions such as prioritization.

Priority Queuing

This topic describes the priority queuing algorithm.



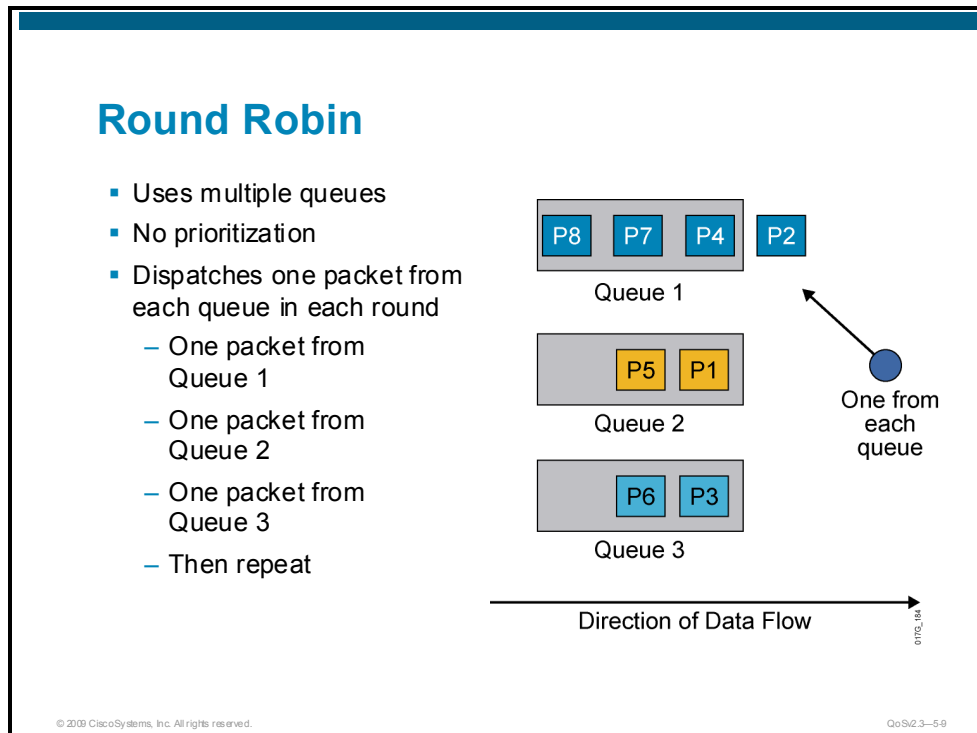
The PQ algorithm is also quite simple. Each packet is assigned a priority and placed into a hierarchy of queues based on priority. When there are no more packets in the highest queue, the next lower queue is serviced. Packets are then dispatched from the next highest queue until either the queue is empty or another packet arrives for a higher PQ.

Packets will be dispatched from a lower queue only when all higher priority queues are empty. If a packet arrives for a higher queue, the packet from the higher queue is dispatched before any packets in lower-level queues.

The problem with PQ is that queues with lower priority can “starve” if a steady stream of packets continues to arrive for a queue with a higher priority. Packets waiting in the lower priority queues may never be dispatched.

Round Robin

This topic describes the round-robin queuing algorithm.



With round-robin queuing, one packet is taken from each queue and then the process repeats.

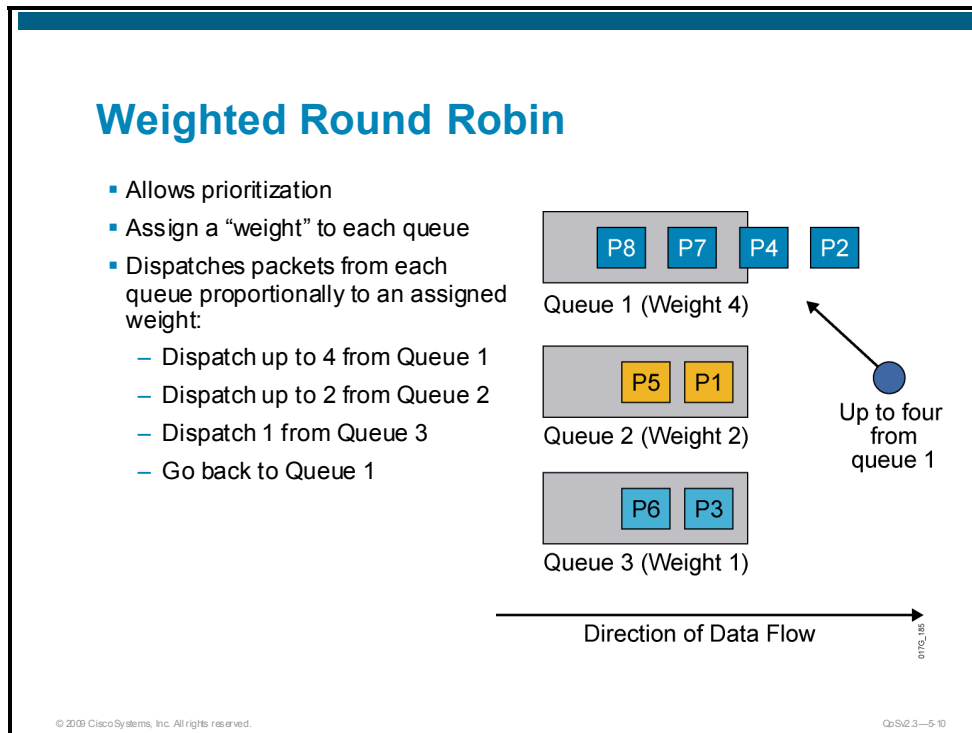
If all packets are the same size, all queues share the bandwidth equally. If packets being put into one queue are larger, that queue will receive a larger share of bandwidth.

No queue will “starve” with round robin because all queues receive an opportunity to dispatch a packet every round.

A limitation of round robin is the inability to prioritize traffic.

Weighted Round Robin

This topic describes the weighted round robin (WRR) queuing algorithm.



The WRR algorithm was developed to provide prioritization capabilities for round robin.

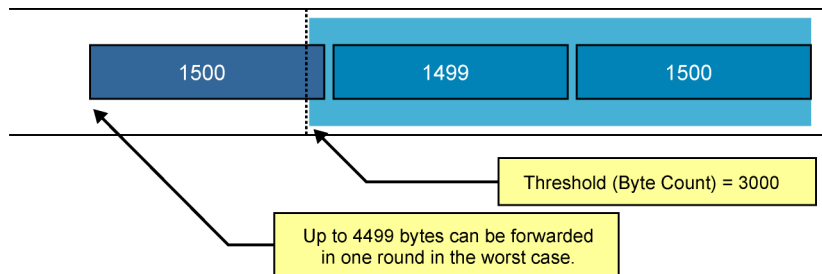
In WRR, packets are assigned a class (voice, file transfer, and so on) and placed into the queue for that class of service. Packets are accessed round-robin style, but queues can be given priorities called “weights.” For example, in a single round, four packets from a high-priority class might be dispatched, followed by two from a middle-priority class, and then one from a low-priority class.

Some implementations of the WRR algorithm will dispatch a configurable number of bytes during each round.

Weighted Round Robin (Cont.)

Problem with WRR

- Some implementations of WRR dispatch a configurable number of bytes (threshold) from each queue for each round—several packets can be sent in each turn.
- The router is allowed to send the entire packet even if the sum of all bytes is more than the threshold.



Some implementations of the weighted round-robin algorithm provide preferential service by dispatching a configurable number of bytes each round rather than a number of packets. The Cisco custom queuing (CQ) mechanism is an example of this implementation.

The figure illustrates a unique-case scenario of the WRR algorithm, in which these parameters were used to implement WRR queuing on an interface:

- Maximum transmission unit (MTU) of the interface is 1500 bytes.
- The byte count to be sent for the queue in each round is 3000 bytes (twice the MTU).

The example shows how the router first sent two packets with a total size of 2999 bytes. Because this is still within the limit (3000), the router can send the next packet (MTU-sized). The result was that the queue received almost 50 percent more bandwidth in this round than it should have received.

This is one of the drawbacks of WRR queuing—it does not allocate bandwidth accurately.

The limit or weight of the queue is configured in bytes. The accuracy of WRR queuing depends on the weight (byte count) and the MTU. If the ratio between the byte count and the MTU is too small, WRR queuing will not allocate bandwidth accurately. If the ratio between the byte count and the MTU is too large, WRR queuing will cause long delays.

Deficit Round Robin

This topic describes the deficit round-robin queuing algorithm.

Deficit Round Robin

- Solves problem with some implementations of WRR
- Keeps track of the number of “extra” bytes dispatched in each round—the “deficit”
- Adds the “deficit” to the number of bytes dispatched in the next round
- Problem resolved with deficit round robin:
 - Threshold of 3000
 - Packet sizes of 1500, 1499, and 1500
 - Total sent in round = 4499 bytes
 - Deficit = $(4499 - 3000) = 1499$ bytes
 - On the next round send only the $(\text{threshold} - \text{deficit}) = (3000 - 1499) = 1501$ bytes

© 2009 Cisco Systems, Inc. All rights reserved.

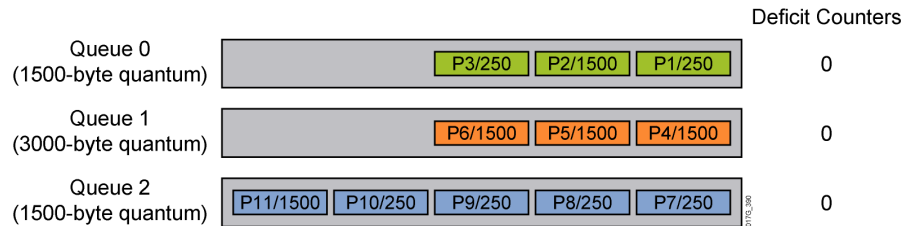
QoS/2.3—5.12

Deficit round robin is an implementation of the WRR algorithm developed to resolve the inaccurate bandwidth allocation problem with WRR. The Cisco modified deficit round robin (MDRR) method used on the Cisco 12000 series is an implementation of deficit round robin.

Deficit round robin uses a deficit counter to track the number of “extra” bytes dispatched over the number of bytes that was to be configured to be dispatched each round. During the next round, the number of “extra” bytes—the deficit—is effectively subtracted from the configurable number of bytes that are dispatched.

Modified Deficit Round Robin

- Differs from regular DRR by adding a special low-latency queue that can be serviced in one of two modes:
 - Strict priority mode: Low-latency queue is serviced whenever it is not empty.
 - Alternate mode: MDRR alternately services the low-latency queue and any other configured queues.
- Each queue within MDRR is defined by:
 - Quantum value: Average number of bytes served in each round.
 - Deficit counter: Number of bytes a queue has transmitted in each round.



MDRR is a class-based composite scheduling mechanism that allows for queueing of up to eight traffic classes. It operates in the same manner as class-based weighted fair queueing (CBWFQ) and allows definition of traffic classes based on customer match criteria (such as access lists); however, MDRR does not use the weighted fair queueing algorithm.

With MDRR configured in the queueing strategy, nonempty queues are served one after the other, in a round-robin fashion. Each time a queue is served, a fixed amount of data is dequeued. The algorithm then services the next queue. When a queue is served, MDRR keeps track of the number of bytes of data that were dequeued in excess of the configured value. In the next pass, when the queue is served again, less data is dequeued to compensate for the excess data that was served previously. As a result, the average amount of data dequeued per queue is close to the configured value.

Each queue within MDRR is defined by these two variables:

- **Quantum value:** Average number of bytes served in each round.
- **Deficit counter:** Number of bytes a queue has transmitted in each round. The counter is initialized to the quantum value.

Packets in a queue are served as long as the deficit counter is greater than zero. Each packet served decreases the deficit counter by a value equal to its length in bytes. A queue can no longer be served after the deficit counter becomes zero or negative. In each new round, the deficit counter for each nonempty queue is incremented by its quantum value. In general, the quantum size for a queue should not be smaller than the maximum transmission unit (MTU) of the interface to ensure that the scheduler always serves at least one packet from each nonempty queue.

Each MDRR queue can be given a relative weight, with one of the queues in the group defined as a priority queue. The weights assign relative bandwidth for each queue when the interface is congested. The MDRR algorithm dequeues data from each queue in a round-robin fashion if there is data in the queue to be sent. During each cycle, a queue can dequeue a quantum based on its configured weight.

MDRR differs from regular DRR by adding a special low-latency queue that can be serviced in one of two modes:

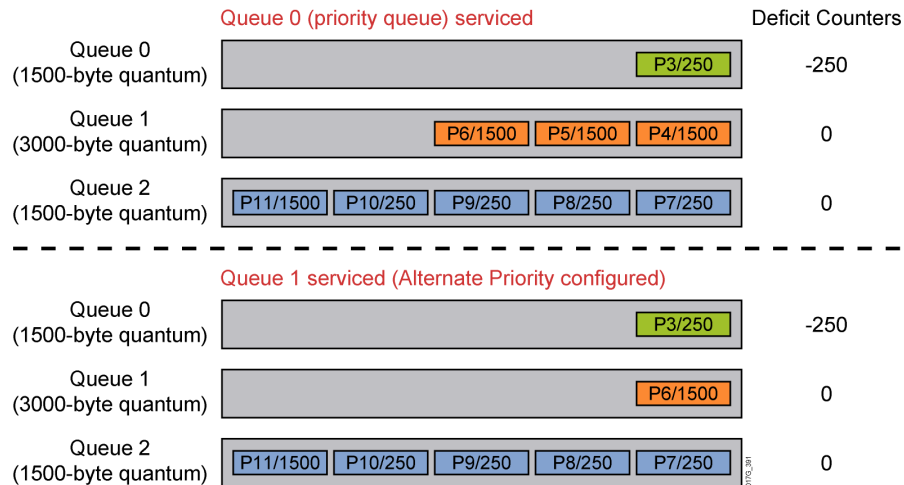
- **Strict priority mode:** The low-latency queue is serviced whenever it is not empty. This provides the lowest delay possible for delay-sensitive traffic. The scheduler services only the current non-priority packet and then switches to the low-latency queue. The scheduler starts to service a non-priority queue only after the low-latency queue becomes completely empty. This mode can starve other queues, particularly if the matching flows are aggressive senders.
- **Alternate mode:** The MDRR scheduler alternatively services the low-latency queue and any other configured queues. Alternate mode can exercise less control over jitter and delay. If the MDRR scheduler starts to service frames from a data queue and then a voice packet arrives in the low-latency queue, the scheduler completely serves the non-priority queue until its deficit counter reaches zero. During this time, the low-latency queue is not serviced, and the VoIP packets are delayed. It is important to note that the priority queue in alternate priority mode is serviced more than once in a cycle, and thus takes more bandwidth than other queues with the same nominal weight. How much more is a function of how many queues are defined. For example, with three queues, the low latency queue is serviced twice as often as the other queues, and it sends twice its weight per cycle.

The figure shows three queues, each of which contains some packets that have been received and queued. For example, Queue 0 contains three packets: P1 (a 250-byte packet), P2 (a 1500-byte packet), and P3 (another 250-byte packet). Queue 0 is the low latency queue, and it is configured to operate in alternate mode. Each queue is assigned a quantum, as follows:

- Queue 0 has a quantum of 1500 bytes.
- Queue 1 has a quantum of 3000 bytes.
- Queue 2 has a quantum of 1500 bytes.

The figure illustrates the initial state of the queues for an example of MDRR operations, which is continued on the following pages.

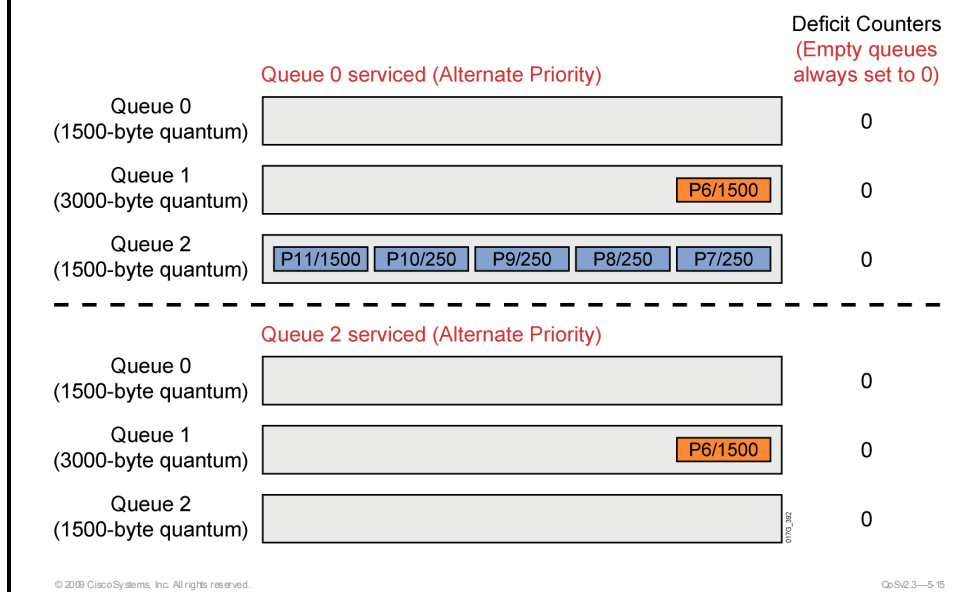
Modified Deficit Round Robin (Cont.)



Queue 0 is serviced first. Its quantum is added to its deficit counter, Packet 1 (250 bytes) is transmitted, and its size is subtracted from the deficit counter. Because the deficit counter of Queue 0 is still greater than 0 ($1500 - 250 = 1250$), Packet 2 is transmitted as well, and its length subtracted from the deficit counter. The deficit counter of Queue 0 is now -250, so queue 1 is serviced next.

The deficit counter of Queue 1 is set to 3000, and packets 4 and 5 are transmitted. As each packet is transmitted, its size is subtracted from the deficit counter. This reduces the deficit counter of Queue 1 to 0.

Modified Deficit Round Robin (Cont.)



Because alternate priority mode is configured, MDRR services Queue 0 again after servicing Queue 1. Again, the Queue 0 quantum value of 1500 is added to the current deficit counter of -250, making the new deficit counter of Queue 0 1250. Packet 3 (whose size is 250 bytes) is now transmitted, and Queue 0 is now empty. When a queue is emptied, its deficit counter is set to 0, as shown in the figure.

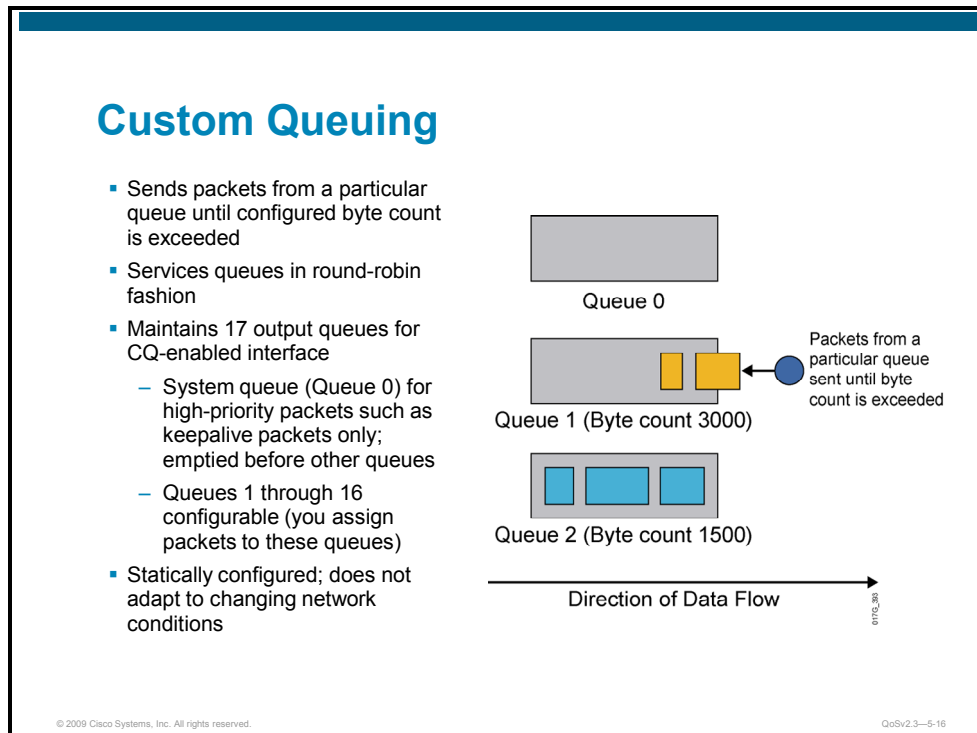
Queue 2 is serviced next. Its deficit counter is set to 1500. Packets 7 through 10 are transmitted, which leaves the deficit counter at 500 ($1500 - (4 \times 250) = 500$). Because the deficit counter is still greater than 0, packet 11 is also transmitted. When packet 11 is transmitted, Queue 2 is empty, and its deficit counter is set to 0, as shown in the figure.

Note MDRR is supported in Cisco IOS XR Software. For more information on MDRR, please refer to the document *Understand and Configure MDRR/WRED on the Cisco 12000 Series Internet Router* and the section "Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software" in the *Cisco IOS XR Modular Quality of Service Configuration Guide, Release 3.8*, on Cisco.com.

Note WFQ, CBWFQ, and LLQ are explained in depth in another lesson.

Custom Queuing

This topic describes the custom queuing algorithm.



The Cisco IOS QoS CQ feature can provide guaranteed bandwidth to specific traffic at a potential congestion point, thus assuring the traffic a fixed portion of available bandwidth and leaving the remaining bandwidth to other traffic. For example, you could reserve half of the bandwidth for Systems Network Architecture (SNA) data, allowing the remaining half to be used by other protocols. If a particular type of traffic is not using the bandwidth reserved for it, unused bandwidth can be dynamically allocated to other traffic types.

Note CQ is statically configured and does not adapt to changing network conditions. With CQ enabled, the system takes longer to switch packets than FIFO because the packets are classified by the processor card.

CQ allows you to specify a certain number of bytes to forward from a queue each time the queue is serviced. This allows you to share the network resources among applications with specific minimum bandwidth requirements. You can also specify a maximum number of packets in each queue.

CQ handles traffic by specifying the number of packets or bytes to be serviced for each class of traffic. It services the queues by cycling through them in round-robin fashion, sending the portion of allocated bytes for each queue before moving to the next queue. If one queue is empty, the router sends packets from the next queue that has packets ready to send.

When CQ is enabled on an interface, the system maintains 17 output queues for that interface. You can specify queues 1 through 16. Queue number 0 is a system queue; it is emptied before any of the queues numbered 1 through 16 are processed. The system queues high-priority packets, such as keepalive packets and signaling packets, to this queue. Other traffic cannot be configured to use this queue.

In order to allocate bandwidth to different queues, you must specify the byte count for each queue. The router sends packets from a particular queue until the byte count is exceeded. Once the byte count value is exceeded, the packet that is currently being sent is completely sent. Therefore, if you set the byte count to 100 bytes and the packet size of your protocol is 1024 bytes, then every time this queue is serviced, 1024 bytes will be sent rather than 100 bytes.

For example, suppose one protocol has 500-byte packets, another has 300-byte packets, and a third has 100-byte packets. If you want to split the bandwidth evenly across all three protocols, you might choose to specify byte counts of 200, 200, and 200 for each queue. However, this configuration does not result in a 33/33/33 ratio. When the router services the first queue, it sends a single 500-byte packet; when it services the second queue, it sends a 300-byte packet; and when it services the third queue, it sends two 100-byte packets. The effective ratio is 50/30/20. Thus, setting the byte count too low can result in an unintended bandwidth allocation.

However, very large byte counts may also cause an unacceptable distribution. That is, if you assign 10 KB, 10 KB, and 10 KB to the three queues in the example, each protocol is serviced promptly when its queue is the one being serviced, but it may be a long time before the queue is serviced again. In order to service queues in a timely manner and ensure that the configured bandwidth allocation is as close as possible to the required bandwidth allocation, you must determine the byte count based on the packet size of each protocol. Otherwise your percentages may not match what you configure.

Note	CQ was modified in Cisco IOS Software Release 12.1. When the queue is depleted early, or the last packet from the queue does not exactly match the configured byte count, the amount of deficit is remembered and accounted for the next time the queue is serviced. Beginning with Cisco IOS Software Release 12.1, you need not be as accurate in specifying byte counts as you did when using earlier Cisco IOS Software releases that did not take deficit into account.

Window size also affects the bandwidth distribution. If the window size of a particular protocol is set to one, that protocol will not place another packet into the queue until it receives an acknowledgment. The CQ algorithm moves to the next queue if the byte count is exceeded or no packets are in that queue.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Congestion can occur at any point in the network, but particularly at points of speed mismatches and traffic aggregation.
- Three basic queuing algorithms are used to manage congestion: FIFO, priority, and round-robin queuing.
- FIFO is the simplest queuing algorithm.
- Priority queuing allows for the prioritization of traffic through the use of multiple queues, but can starve lower-priority queues.
- Round-robin queuing uses multiple queues to provide equal access to all queues.
- Weighted round robin offers priority access to multiple queues by assigning “weights” to queues, but some implementations may provide inaccurate access to some queues.
- Deficit round-robin queuing solves the inaccuracy problem with round robin by keeping a “deficit” count.
- Custom queuing enables you to guarantee a fixed portion of available bandwidth to specific traffic and leave the remaining bandwidth to other traffic.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3--5-17

Understanding Queuing Implementations

Overview

Queuing technologies are one of the primary ways to manage congestion in a network. Network administrators and network designers benefit from an understanding of how the queuing algorithms are implemented. This lesson describes the relationship of queuing components and how this affects the overall queuing performance.

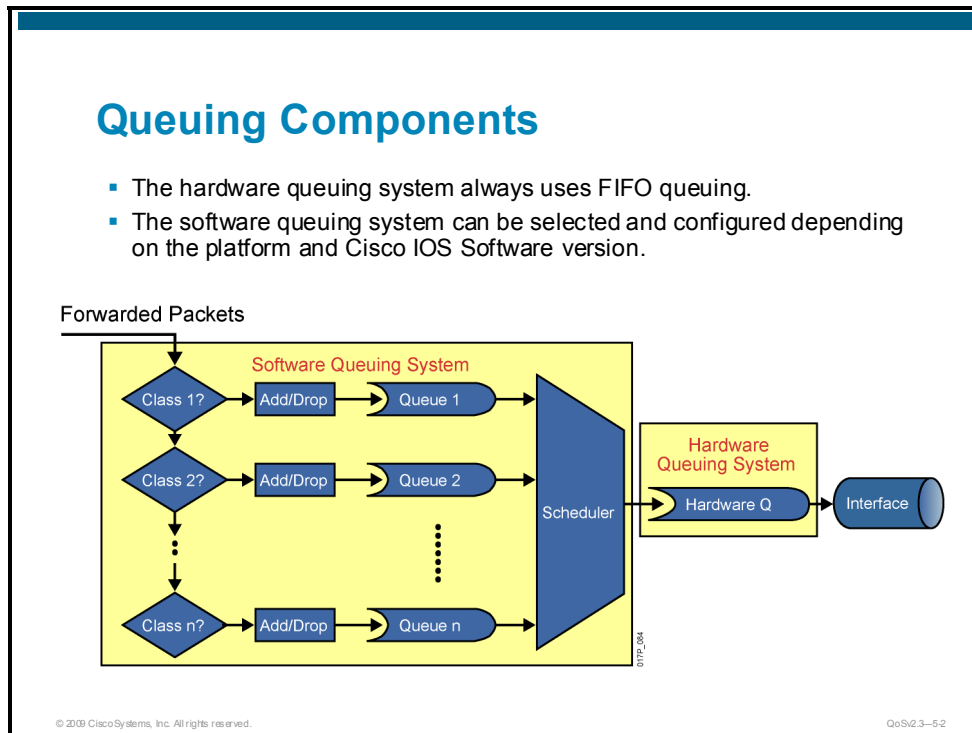
Objectives

Upon completing this lesson, you will be able to describe hardware and software queuing on a network device. This ability includes being able to meet these objectives:

- Describe the primary components of a queuing mechanism
- Explain the significance of the size of the hardware queue
- Explain how congestion occurs on software interfaces
- Describe the basic software queuing technologies used on Cisco network devices

Queuing Components

This topic describes the primary components of a queuing mechanism.



Queuing on routers is necessary to accommodate bursts when the arrival rate of packets is greater than the departure rate, usually because of one of these two reasons:

- Input interface is faster than the output interface
- Output interface is receiving packets coming in from multiple other interfaces

Initial implementations of queuing used a single FIFO strategy. More complex scheduling mechanisms were introduced when special requirements needed routers to differentiate between packets of different levels of importance.

Queuing was split into these two parts:

- **Hardware queue:** Uses FIFO strategy, which is necessary for the interface drivers to transmit packets one by one. The hardware queue is sometimes referred to as the transmit queue (TxQ).
- **Software queue:** Schedules packets into the hardware queue based on the QoS requirements.

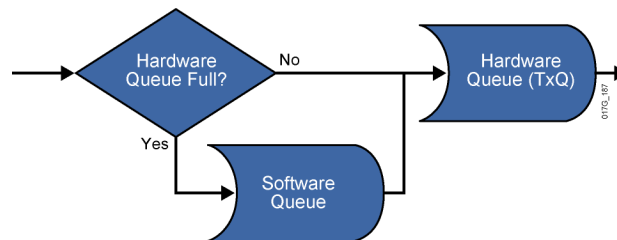
The figure illustrates these actions that have to be taken before a packet can be transmitted:

- Most queuing mechanisms include classification of packets.
- After a packet is classified, a router has to determine whether it can put the packet into the queue or drop the packet. Most queuing mechanisms will drop a packet only if the corresponding queue is full (tail drop). Some mechanisms use a more intelligent dropping scheme, such as weighted fair queuing (WFQ), or a random dropping scheme, such as weighted random early detection (WRED).

- If the packet is allowed to be enqueued, it will be put into the FIFO queue for that particular class.
- Packets are then taken from the individual per-class queues and put into the hardware queue.

The Software Queue

- Generally, a **full hardware queue** indicates **interface congestion**, and software queuing is used to manage it.
- When a packet is being forwarded, the router will bypass the software queue if the hardware queue has space in it (no congestion).



© 2009 Cisco Systems, Inc. All rights reserved.

QoS2.3-5.3

Software queuing is implemented when the interface is congested; the software queuing system is bypassed whenever there is room in the hardware queue.

The software queue is, therefore, used only when data must wait to be placed into the hardware queue.

Hardware Queue (TxQ) Size

This topic describes the significance of the size of the hardware queue.

Hardware Queue (TxQ) Size

- Routers determine the length of the hardware queue based on the configured bandwidth of the interface.
- The length of the hardware queue can be adjusted with the **tx-ring-limit** command.
- Reducing the size of the transmit ring has two benefits:
 - It reduces the maximum amount of time that packets wait in the FIFO queue before being transmitted.
 - It accelerates the use of QoS in the Cisco IOS Software.
- Improper tuning of the hardware queue may produce undesirable results:
 - **Long TxQ** may result in poor performance of the software queue.
 - **Short TxQ** may result in a large number of interrupts, which cause high CPU utilization and low link utilization.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.4

The double queuing strategy (software and hardware queue) has its impacts on the results of overall queuing. Software queues serve a valuable purpose. If the hardware queue is too long, it will contain a large number of packets scheduled in the FIFO fashion. A long FIFO hardware queue most likely defeats the purpose of the QoS design requiring a certain complex software queuing system (for example, custom queuing [CQ]).

So why use the hardware queue at all? Or why not just set its length to one? That would force all packets to go through the software queue and be scheduled one by one to the interface for transmission. This approach has these drawbacks:

- Each time a packet is transmitted, the interface driver interrupts the CPU and requests more packets to be delivered into its hardware queue. Some queuing mechanisms have complex scheduling that takes time to deliver more packets. The interface does not send anything during that time (link utilization is decreased) if the hardware queue is empty, because its maximum size is one.
- The CPU schedules packets one by one instead of many at the same time (in the same interrupt interval), which increases the CPU utilization.

The default TxQ size is determined by the Cisco IOS Software, based on the bandwidth of the media, and should be fine for most queuing implementations. Some platforms and QoS mechanisms will automatically adjust the TxQ size to an appropriate value. Faster interfaces have longer hardware queues because they produce less delay. Slower interfaces have shorter hardware queues to prevent too much delay in the worst-case scenario in which the entire hardware queue is full of MTU-sized packets.

Note Refer to Cisco IOS Software configuration documentation for more information.

The transmit ring serves as a staging area for packets in line to be transmitted. The router needs to enqueue a sufficient number of packets on the transmit ring and ensure that the interface driver has packets with which to fill available cell timeslots.

The primary reason to tune the transmit ring is to reduce latency caused by queuing. On any network interface, queuing forces a choice between latency and the amount of burst that the interface can sustain. Larger queue sizes sustain longer bursts while increasing delay. Tune the size of a queue when traffic is experiencing unnecessary delay.

The size of the transmit ring must be small enough to avoid introducing latency because of queuing. The transmit ring must also be large enough to avoid drops and a resulting impact to TCP-based flows. Queuing on the transmit ring introduces a serialization delay that is directly proportional to the depth of the ring. An excessive serialization delay can impact latency budgets for delay-sensitive applications such as voice. Cisco recommends reducing the size of the transmit ring for virtual circuits (VCs) carrying voice. Select a value based on the amount of serialization delay, expressed in seconds, introduced by the transmit ring. Use the following formula:

$$((P * 8) * D) / S$$

P = Packet size in bytes. Multiply by eight to convert to bits.

D = Transmit-ring depth.

S = Speed of the VC in bps.

Note IP packets on the Internet are typically one of three sizes: 64 bytes (for example, control messages), 1500 bytes (for example, file transfers), or 256 bytes (all other traffic). These values produce a typical overall Internet packet size of 250 bytes.

Congestion on Software Interfaces

This topic describes how congestion occurs on software interfaces.

Congestion on Software Interfaces

- Subinterfaces and software interfaces do not have their own separate Tx ring, therefore no congestion can occur.
 - Dialers, tunnels, Frame Relay subinterfaces
 - They congest when their main hardware interface Tx ring congests
- The **Tx-ring state** (full, not full) is, therefore, an indication of congestion for software interfaces.
- Only hardware interfaces have a Tx ring.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.5

Subinterfaces and software interfaces do not have their own separate transmit (Tx) ring; therefore, no congestion can occur. These interface types include dialers, tunnels, and Frame Relay subinterfaces, and will only congest when their main hardware interface Tx ring congests. The Tx ring state is an indication of congestion for software interfaces.

Software interfaces can implement a software shaping queue when traffic shaping is configured on the Frame Relay subinterface, and, in this case, the main interface software queue will be a FIFO queue.

Queuing Implementations in Cisco IOS Software

This topic describes the basic software queuing technologies used on Cisco network devices.

Queuing Implementations in Cisco IOS Software

- Priority queuing
 - Implementation of priority queuing
 - Four queues (high, medium, normal, low)
- Custom queuing
 - Implementation of weighted round robin
 - Up to 16 queues
 - Threshold based on number of bytes
 - Configurable priority queues
 - Inaccurate bandwidth allocation due to threshold issue with weighted round robin
- Modified deficit round robin
 - Deficit round robin with a priority queue for Cisco 12xxx routers

© 2003 Cisco Systems, Inc. All rights reserved.

QoS2.3-5.6

The figure shows these available software queuing technologies:

- Priority queuing (PQ):
 - A Cisco implementation of the priority queuing algorithm
 - Allows four queues to be used for prioritization (high, medium, normal, low)
 - Allows for a variety of classification, including source IP address, destination IP address, IP precedence, and differentiated services code point (DSCP)
- Custom queuing (CQ):
 - A Cisco implementation of weighted round robin (WRR)
 - Allows up to 16 queues to be used for traffic classification
 - Allows for a variety of classification, including source IP address, destination IP address, IP Precedence, and DSCP
 - Tail drop used within each individual queue
- Modified deficit round robin (MDRR):
 - A Cisco implementation of deficit round robin
 - Available only on the Cisco 12000 Series routers

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Each physical interface has a hardware and a software queuing system. The hardware queuing system uses FIFO, while the software queuing system can be configured depending on the platform and IOS version.
- The length of the hardware queue has a significant impact on performance and can be configured on a router with the **tx-ring-limit** command.
- Software interfaces have no queues; they congest only when their hardware interface congests.
- Cisco offers implementations of basic queuing algorithms: priority queuing, custom queuing, and modified deficit round robin.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS/2.3-5.7

Configuring FIFO and WFQ

Overview

FIFO and weighted fair queuing (WFQ) are the two primary default queuing mechanisms that are implemented on Cisco routers. WFQ was developed to resolve some of the problems, such as queue starvation, delay, and jitter, resulting from the use of basic queuing methods. WFQ dynamically divides available bandwidth by a calculation based on the total number of flows and the weight of each given flow. Bandwidth cannot be guaranteed, because the number of flows are constantly changing and therefore so is the allocated bandwidth to each flow. This lesson describes how to configure WFQ to manage congestion.

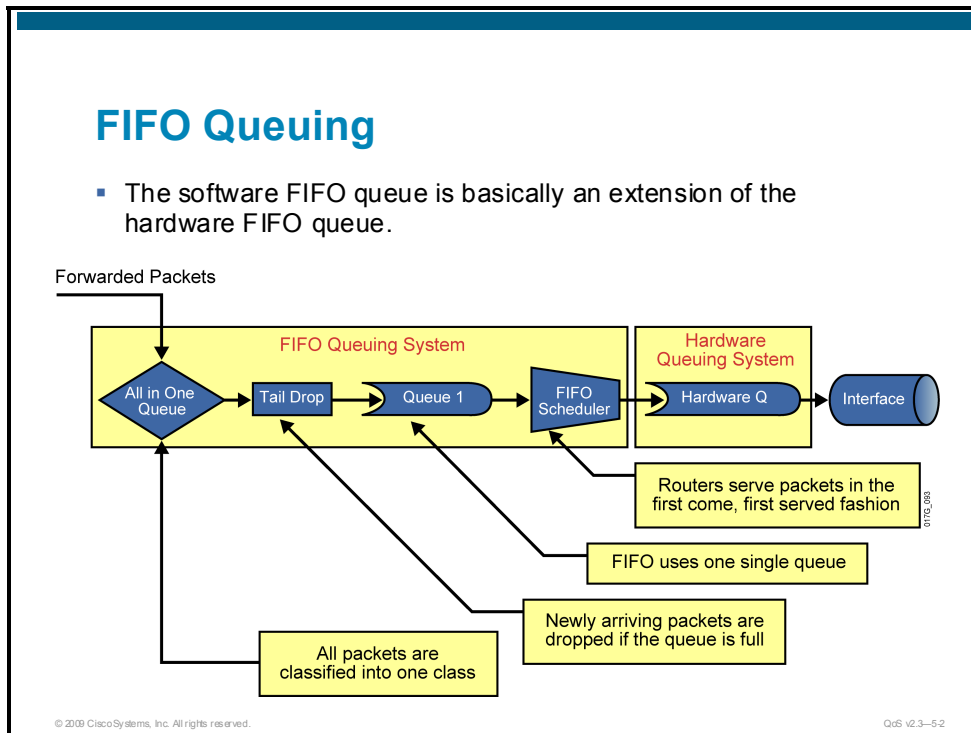
Objectives

Upon completing this lesson, you will be able to configure WFQ to manage congestion, given a network with suboptimal QoS performance. This ability includes being able to meet these objectives:

- Describe the FIFO queuing mechanism
- Give a detailed explanation of WFQ
- Identify the parameters on which WFQ can classify traffic
- Explain the insertion and drop policy used by WFQ
- Explain how finish time is calculated based on weight and how it is used in the operation of WFQ
- Describe the benefits and drawbacks of using WFQ to implement QoS
- Identify the Cisco IOS commands required to configure WFQ on a Cisco router
- Identify the Cisco IOS commands required to monitor WFQ on a Cisco router

FIFO Queuing

This topic describes the FIFO queuing mechanism.



FIFO queuing has no classification because all packets belong to the same class. Packets are dropped when the output queue is full (tail drop). The scheduler services packets in the order they arrive.

The software FIFO queue is basically an extension of the hardware FIFO queue.

FIFO Queuing (Cont.)

- **Benefits**
 - **Simple** and **fast** (one single queue with a simple scheduling mechanism)
 - Supported on all platforms
 - Supported in all switching paths
 - Supported in all IOS versions
- **Drawbacks**
 - Causes **starvation** (aggressive flows can monopolize links)
 - Causes **jitter** (bursts or packet trains temporarily fill the queue)

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.3

Although FIFO queuing might be regarded as the fairest queuing mechanism, it has these drawbacks:

- FIFO is extremely unfair when an aggressive flow contests with a fragile flow. Aggressive flows send a large number of packets, many of which are dropped. Fragile flows send a modest number of packets and most of them are dropped because the queue is always full due to the aggressive flow. This type of behavior is called starvation.
- Short or long bursts cause a FIFO queue to fill. Packets entering an almost-full queue have to wait a long time before they can be transmitted. At another time, the queue might be empty, causing packets of the same flow to experience almost no delay. Variation in delay is called jitter.

Despite the drawbacks, FIFO is still the most-used queuing mechanism because of these benefits:

- FIFO is simple and fast. Most high-end routers with fast interfaces are not really challenged by the drawbacks. Furthermore, routers are not capable of complex classification and scheduling when they have to process a large number of packets per second. FIFO is, therefore, the most suitable queuing mechanism on high-end router platforms.
- FIFO is supported on all platforms.
- FIFO queuing is supported in all versions of Cisco IOS Software.

Weighted Fair Queuing

This topic describes weighted fair queuing (WFQ).

Weighted Fair Queuing

- A queuing algorithm should share the bandwidth **fairly** among flows by:
 - Reducing response time for interactive flows by scheduling them to the front of the queue
 - Preventing high-volume conversations from monopolizing an interface
- In the WFQ implementation, messages are sorted into **conversations (flows)** and transmitted by the order of the last bit crossing its channel.
- Unfairness is reinstated by introducing **weight** to give proportionately more bandwidth to flows with higher IP precedence (lower weight).

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3–6.4

WFQ was introduced as a solution to these problems of the queuing mechanisms:

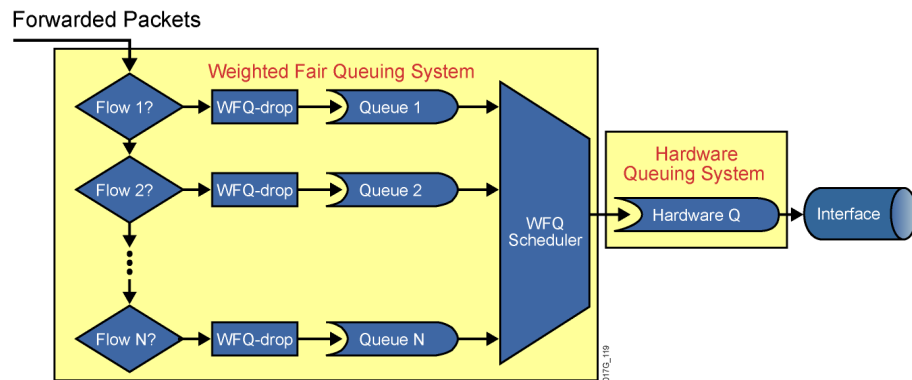
- FIFO queuing causes starvation, delay, and jitter.
- Priority queuing (PQ) causes starvation of other lower priority classes and suffers from all FIFO problems within each of the four queues that it uses for prioritization.
- Custom queuing (CQ) causes long delays and also suffers from all FIFO problems within each of the 16 queues that it uses for traffic classification.

The idea of WFQ is as follows:

- To have a dedicated queue for each flow (no starvation, delay, or jitter within the queue)
- To allocate bandwidth fairly and accurately among all flows (minimum scheduling delay, guaranteed service)
- To use IP precedence as weight when allocating bandwidth

WFQ Architecture

- WFQ uses **per-flow** FIFO queues



WFQ uses automatic classification. Manually defined classes are not supported.

WFQ dropping is not a simple tail drop. WFQ drops packets of the most aggressive flows.

WFQ scheduler is a simulation of a time-division multiplexing (TDM) system. WFQ adapts to the number of active flows (queues) and attempts to allocate equal amounts of bandwidth to each flow (queue). Flows with small packets (usually interactive flows) get much better service because they do not need a great deal of bandwidth. They need low delay, however, which they get because of the way in which the WFQ algorithm works.

WFQ Implementations

- Implementation parameters
 - Queuing platform: central CPU or VIP
 - Classification mechanism
 - Weighted fairness
- **Modified tail drop** within each queue

© 2009 Cisco Systems, Inc. All rights reserved.

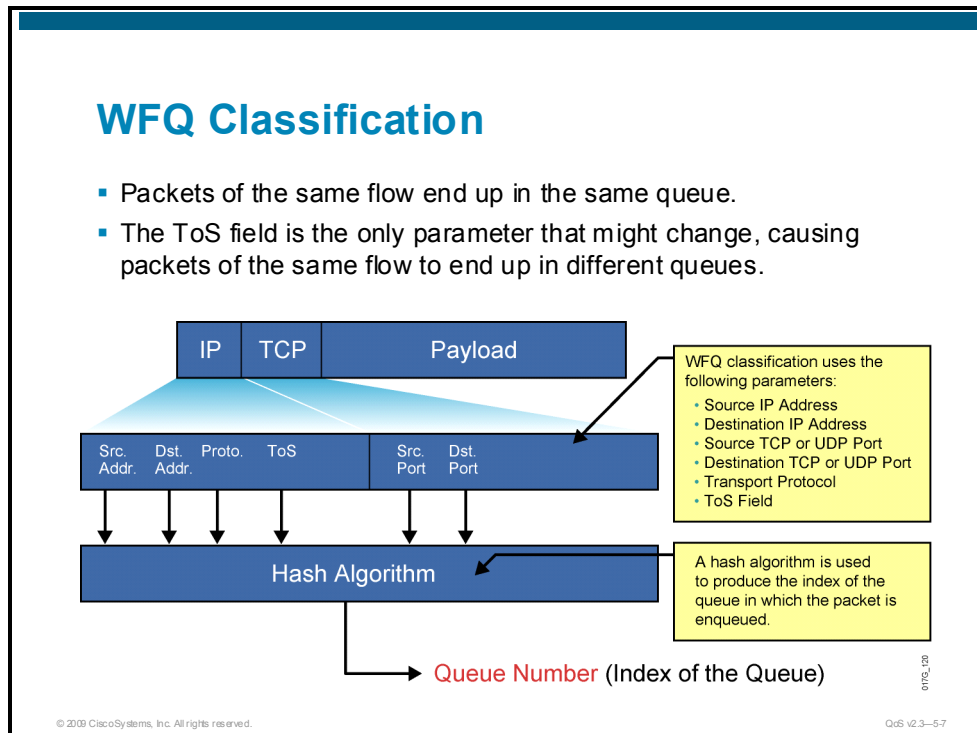
QoS v2.3-5.6

WFQ is supported on most Cisco routers, as well as on Versatile Interface Processors (VIPs). The implementation of WFQ on the VIP is distinct in these ways:

- Classification identifies a flow and assigns a queue to the flow.
- Weight is used for scheduling to give proportionately more bandwidth to flows with a higher IP precedence.
- The tail-dropping scheme is improved to drop packets of the most aggressive flows.

WFQ Classification

This topic describes the parameters on which WFQ can classify traffic.



WFQ classification has to identify individual flows. (The term “conversation” is also used to signify flow.) A flow is identified based on information taken from the IP header and the TCP or User Datagram Protocol (UDP) headers, such as the following:

- Source IP address
- Destination IP address
- Protocol number (identifying TCP or UDP)
- Type of service field
- Source TCP or UDP port number
- Destination TCP or UDP port number

These parameters are usually fixed for a single flow, although there are some exceptions:

- A QoS design can mark packets with different IP precedence values even if they belong to the same flow. You should avoid this when using WFQ.
- Some applications change port numbers (for example, TFTP).

If packets of the same flow do not have the same parameters (for example, a different ToS field), the packets can end up in different queues and reordering can occur.

The parameters are used as input for a hash algorithm that produces a fixed-length number that is used as the index of the queue.

WFQ Classification (Cont.)

- A fixed number of per-flow queues is configured.
- A hash function is used to translate flow parameters into a queue number.
- System packets (8 queues) and RSVP flows (if configured) are mapped into separate queues.
- Two or more flows could map into the same queue, resulting in lower per-flow bandwidth.
- Important: The number of queues configured has to be larger than the expected number of flows.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.8

WFQ uses a fixed number of queues. The hash function is used to assign a queue to a flow. There are eight additional queues for system packets and optionally up to 1000 queues for Resource Reservation Protocol (RSVP) flows. The number of dynamic queues WFQ uses by default is based on the interface bandwidth. Using the default interface bandwidth, WFQ uses 256 dynamic queues. The number of queues can be configured in the range between 16 and 4096 (the number must be a power of 2). The default number of dynamic queues for different interface bandwidths is shown in the table.

Bandwidth Range	Number of Dynamic Queues
Less than or equal to 64 kb/s	16
More than 64 kb/s and less than or equal to 128 kb/s	32
More than 128 kb/s and less than or equal to 256 kb/s	64
More than 256 kb/s and less than or equal to 512 kb/s	128
More than 512 kb/s	256

If there are a large number of concurrent flows, it is likely that two flows could end up in the same queue. You should have several times as many queues as there are flows (on average). This may not be possible in larger environments where the number of concurrent flows is in the thousands.

The probability of two flows ending up in the same flow could be calculated using the following formula:

$$P = 1 - \frac{Queues!}{Queues^{Flows} \cdot (Queues - Flows)!}$$

The table lists the probability values for three sizes of the WFQ system (64, 128, and 256 queues), with the number of concurrent flows from 5 to 40.

Flows	64 Queues	128 Queues	256 Queues
5	15%	8%	4%
10	52%	30%	16%
15	83%	57%	34%
20	96%	79%	53%
25	100%	92%	70%
30	100%	98%	83%
35	100%	99%	91%
40	100%	100%	96%

The following is the sample calculation of the probability value for 5 flows and 64 queues:

Flows: 5

Queues: 64

$$\begin{aligned}\text{Probability} &= 1 - ((64!) / ((64^5) * (59!))) \\ &= 1 - ((64 * 63 * 62 * 61 * 60) / (64 * 64 * 64 * 64 * 64)) \\ &= 1 - 0.852105618 \\ &= 0.147894382 \text{ or } 14.7\% \text{ (15\% rounded off)}\end{aligned}$$

WFQ Insertion and Drop Policy

This topic describes the WFQ insertion and drop policy.

WFQ Insertion and Drop Policy

- WFQ has two modes of dropping:
 - Early dropping when the congestion discard threshold is reached.
 - Aggressive dropping when the hold-queue out limit is reached.
- WFQ typically drops packets of the most aggressive flow.
- Drop mechanism exceptions:
 - Packet classified into an empty sub-queue is never dropped.
 - Packet precedence has no effect on the dropping scheme.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.9

WFQ uses these two parameters that affect the dropping of packets:

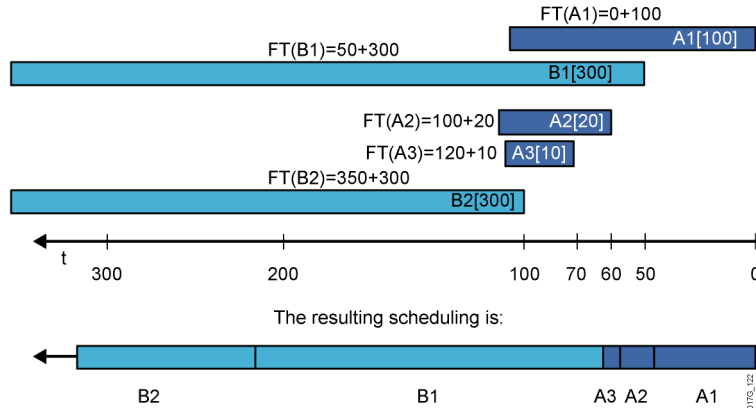
- The congestive discard threshold (CDT) is used to start dropping packets of the most aggressive flow, even before the hold-queue limit is reached.
- The hold-queue out limit (HQP) defines the maximum number of packets that can be in the WFQ system at any time.

There are two exceptions to the WFQ insertion and drop policy, as follows:

- If the WFQ system is above the CDT limit, the packet is still enqueued if the per-flow queue is empty.
- The dropping strategy is not directly influenced by IP precedence.

Finish Time Calculation

If Flow F Active,
Then $FT(P_{k+1}) = FT(P_k) + \text{Size}(P_{k+1})$
Otherwise $FT(P_0) = \text{Now} + \text{Size}(P_0)$



The length of queues (for scheduling purposes) is not in packets but in the time it would take to transmit all the packets in the queue. The end result is that WFQ adapts to the number of active flows (queues) and allocates equal amounts of bandwidth to each flow (queue).

The side effect is that flows with small packets (usually interactive flows) get much better service because they do not need a lot of bandwidth. They need low delay, however, which they get because small packets have a low finish time.

The figure illustrates how two queues (queue A and queue B) are contesting for link bandwidth. For this example, assume that the time units are in ms and time T (value 0 is used in the figure) is the starting point.

Queue A receives packets in the following order and times:

- Packet A1 arrives at time $T + 0$ ms and requires 100 ms to be transmitted.
- Packet A2 arrives at time $T + 60$ ms (the input interface is obviously faster than the output interface because the arrival time of packet A2 is before the finish time of packet A1) and requires 20 ms to be transmitted.
- Packet A3 arrives at time $T + 70$ ms (the input interface is obviously much faster than the output interface) and requires 10 ms to be transmitted.

Queue B receives packets in the following order and times:

- Packet B1 arrives at time $T + 50$ ms and requires 300 ms to be transmitted.
- Packet B2 arrives at time $T + 100$ ms and also requires 300 ms to be transmitted.

The finish times of packets in Queue A are as follows:

- Packet A1 has a finish time that is the sum of the current time (because the queue was empty at the time of arrival) plus the time it takes to transmit this packet (100 ms): $FTA1 = 0 \text{ ms} + 100 \text{ ms} = 100 \text{ ms}$.

- Packet A2 has a finish time that is the sum of the finish time of the last packet in queue A (packet A1) plus the time it would take to transmit this packet (20 ms): $FTA2 = 100 \text{ ms} + 20 \text{ ms} = 120 \text{ ms}$.
- Packet A3 has a finish time that is the sum of the finish time of the last packet in queue A (packet A2) plus the time it would take to transmit this packet (20 ms): $FTA3 = 120 \text{ ms} + 10 \text{ ms} = 130 \text{ ms}$.

The finish times for the packets in Queue B are as follows:

- Packet B1 has a finish time that is the sum of the current time (because the queue was empty at the time of arrival) plus the time it takes to transmit this packet (300 ms): $FTB1 = 50 \text{ ms} + 300 \text{ ms} = 350 \text{ ms}$.
- Packet B2 has a finish time that is the sum of the finish time of the last packet in queue B (packet B1) plus the time it would take to transmit this packet (300 ms): $FTB2 = 350 \text{ ms} + 300 \text{ ms} = 650 \text{ ms}$.

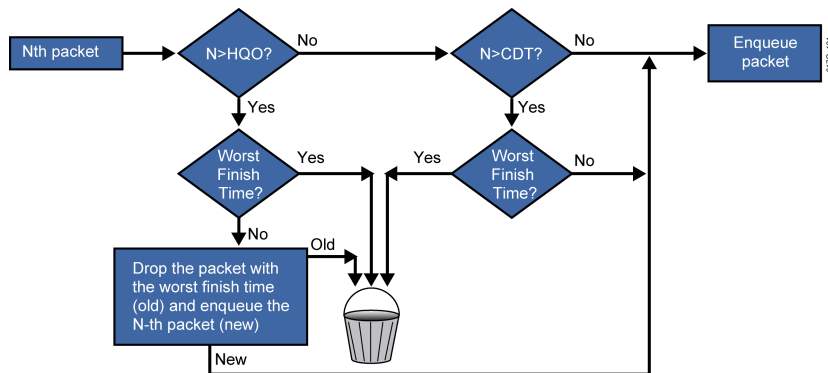
The packets are scheduled into the hardware queue (or the TxQ) in the ascending order of finish times:

1. A1 (100 ms)
2. A2 (120 ms)
3. A3 (130 ms)
4. B1 (350 ms)
5. B2 (650 ms)

Note	WFQ prevents reordering of packets within a single flow (conversation). Small packets are automatically preferred over large packets.

WFQ Dropping Operations

- HQO is the maximum number of packets that the WFQ system can hold.
- CDT is the threshold when WFQ starts dropping packets of the most aggressive flow.
- N is the number of packets in the WFQ system when the Nth packet arrives.

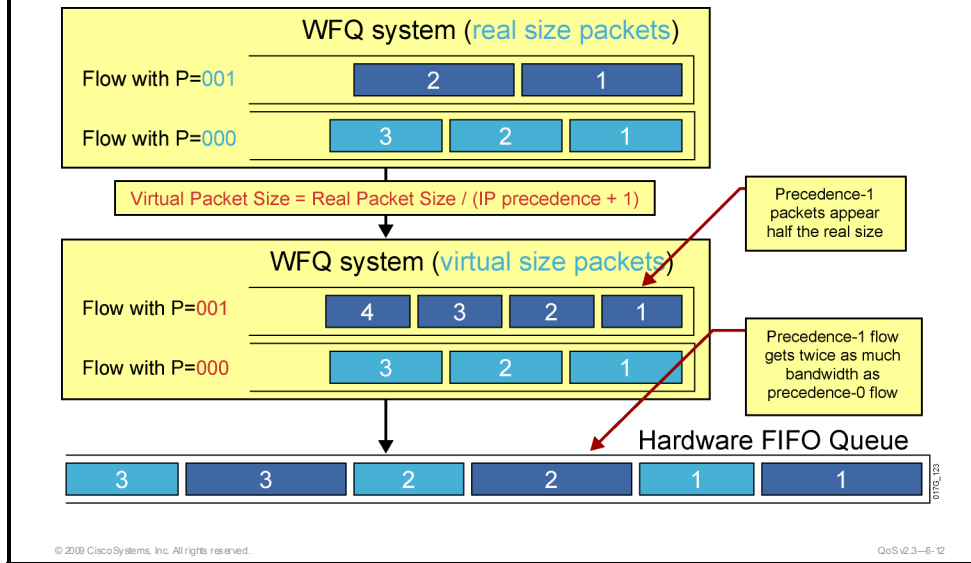


The figure illustrates the dropping scheme of WFQ. The process can be organized into these steps:

- Step 1** Drop the new packet if the WFQ system is full (hold-queue limit reached) and the new packet has the worst finish time (the last in the entire system). WFQ calculates the finish times based on factors such as packet length, packet arrival time, the amount of time needed to transmit the packet, and the finish time of the previous packet placed in the queue.
- Step 2** Drop the packet with the worst finish time in the WFQ system if the system is full. Enqueue the new packet.
- Step 3** When the WFQ system is above the CDT limit, a new packet is dropped if that packet is the last in the WFQ system, even though the WFQ system is still within the hold-queue limit.
- Step 4** When the WFQ system is above the CDT limit, and if a new packet would not be the last in the WFQ system, the new packet can be enqueued and no other packet is dropped.

When the WFQ system is below both the HQO and the CDT limit, newly arriving packets are enqueued normally.

Weight in WFQ Scheduling



The figure introduces the weight into the finish time calculation. The time it takes to transmit the packet is divided by IP Precedence increased by one (to prevent division by zero).

The WFQ implementation in Cisco routers is optimized in these ways:

- The real time it takes to transmit the packet is not relevant. The packet size can be used instead because it is proportional to the transmit time.
- The packet size is not divided by IP precedence (division is a CPU-intensive operation). Instead, the size is multiplied by a fixed value (one multiplication value for each IP Precedence value).

Packets with IP Precedence one appear half the size they really are. The result is that these packets receive twice as much bandwidth as packets with IP Precedence zero.

WFQ Case Study

- WFQ system can hold a maximum of **ten** packets (**hold-queue limit**).
- Early dropping (of aggressive flows) should start when there are **eight** packets (**congestive discard threshold**) in the WFQ system.

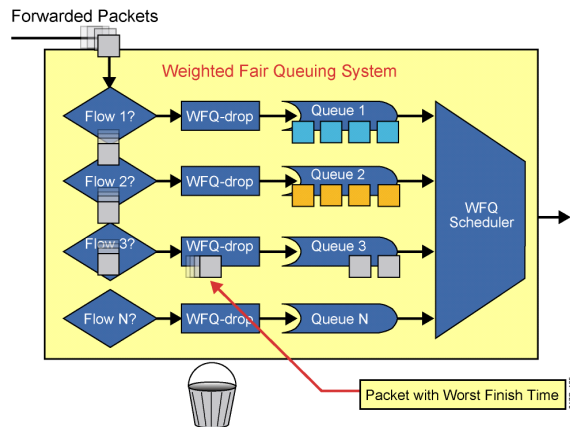
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-13

The case study on the following pages describes how packets are dropped in various situations. The WFQ system is reduced to a modest hold-queue limit of ten and a congestive discard threshold of eight.

WFQ Case Study Interface Congestion

- HQO (hold-queue out limit) is the maximum number of packets that the WFQ system can hold and HQO = 10.



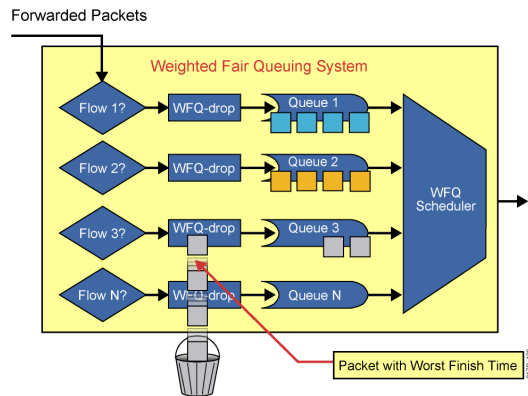
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-14

There are already ten packets in the WFQ system. The new packet would be the eleventh and also the last in the entire WFQ system.

WFQ Case Study Interface Congestion (Cont.)

- **HQO** is the maximum number of packets that the WFQ system can hold and **HQO = 10**.
- Absolute maximum (**HQO=10**) exceeded, new packet is the last in the TDM system and is dropped.



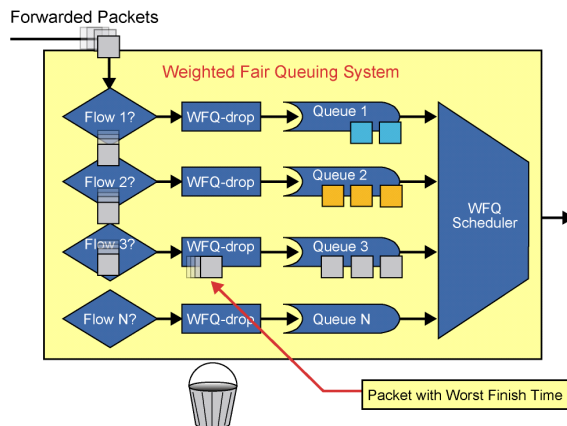
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-15

The new packet is dropped.

WFQ Case Study Flow Congestion

- Early dropping (of aggressive flows) should start when there are **eight** packets (**congestive discard threshold**) in the WFQ system.



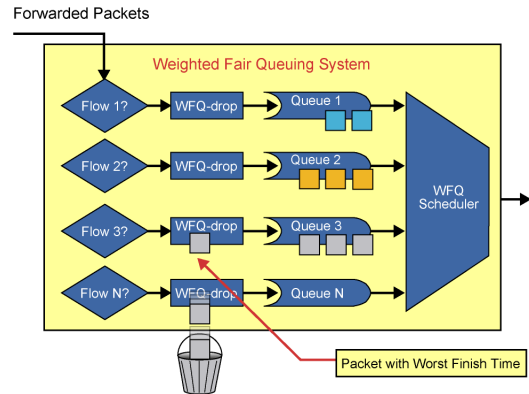
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-16

The example illustrates how WFQ can drop packets even if the WFQ system is still within the hold-queue limit. The system, however, is above the CDT limit. In this case, a packet can be dropped if it is the last in the system.

WFQ Case Study Flow Congestion (Cont.)

- Early dropping (of aggressive flows) should start when there are **eight** packets (**congestive discard threshold**) in the WFQ system.
- CDT exceeded (**CDT=8**), new packet would be the last in the TDM system and is dropped.



© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3-5-17

The new packet would be the last in the TDM system and is dropped.

WFQ Scheduling

This topic describes how finish time is calculated based on weight and how it is used in the operation of WFQ.

Weight in WFQ Scheduling

- WFQ is IP precedence-aware; it can detect higher priority packets marked with precedence values and provide them with superior response time.
- The WFQ scheduler implementation in Cisco routers uses the following information to calculate the finish time:
 - Packet length
 - IP precedence number (translated to a "weight" value)
- The weight value is inversely proportional to the IP precedence value; packets with higher precedence numbers have lower weights.
- Packets with lower weights receive proportionately more bandwidth and priority in the transmit order.
- Packets belonging to RSVP flows and system packets have special low weights that guarantee them more bandwidth.

© 2003 Cisco Systems, Inc. All rights reserved.QoS v2.3-6-18

WFQ is IP precedence-aware. It can detect higher-priority packets marked with precedence values and can schedule them faster, providing superior response time for this traffic.

If IP precedence were not considered, packets would simply be scheduled into the hardware queue (or the TxQ) in the ascending order of finish times. For example, if Packet A had a finish time of 100 ms and Packet B had a finish time of 120 ms, packet A would be placed in the hardware queue first. However, finish time can be adjusted based on IP precedence of the packet.

When determining which packet to transmit next, the WFQ scheduler selects the packet with the lowest finish time among all the queues. The WFQ scheduler uses the following information to calculate the finish time:

- Packet length
- IP precedence number (translated to a "weight" value)

The size of a packet is proportional to the time it takes to transmit the packet. Therefore, the WFQ implementation in Cisco routers uses packet size instead of the time needed to transmit a packet for WFQ finish time calculations. The weight value is inversely proportional to the IP precedence value; packets with higher precedence numbers have lower weights and thus receive proportionately more bandwidth and priority in the transmit order. Packets belonging to RSVP flows and system packets have special low weights that guarantee them more bandwidth.

Note WFQ prevents reordering of packets within a single flow (conversation). Small packets are automatically preferred over large packets.

Benefits and Drawbacks of WFQ

This topic describes the benefits and drawbacks of using WFQ to implement QoS.

Benefits and Drawbacks of WFQ

■ Benefits

- Simple configuration (classification does not have to be configured)
- Guarantees throughput to all flows
- Drops packets of most aggressive flows
- Supported on most platforms
- Supported in all IOS versions

■ Drawbacks

- Multiple flows can end up in one queue
- Does not support the configuration of classification
- Cannot provide fixed bandwidth guarantees
- Complex classification and scheduling mechanisms

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--5-19

These are the main benefits of WFQ:

- Simple configuration (no manual classification is necessary)
- Guarantees throughput to all flows
- Drops packets of the most aggressive flows
- Supported on most platforms
- Supported in all Cisco IOS Software versions

These are the main drawbacks of WFQ:

- Not always possible to have one flow per queue
- Does not allow manual classification
- Cannot provide fixed guarantees
- Complex classification and scheduling mechanisms

Configuring WFQ

This topic describes the Cisco IOS commands required to configure WFQ on a Cisco router.

Configuring WFQ

- **congestive-discard-threshold**
 - Number of messages allowed in the WFQ system before the router starts dropping new packets for the longest queue.
 - The value can be in the range from 1 to 4096 (default is 64)
- **dynamic-queues**
 - Number of dynamic queues used for best-effort conversations (values are: 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096)
- **reservable-queues**
 - Number of reservable queues used for reserved conversations in the range 0 to 1000 (used for interfaces configured for features such as RSVP—the default is 0)

```
router(config-if) #
```

```
fair-queue [congestive-discard-threshold [dynamic-queues  
[reservable-queues]]]
```

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3—6-20

WFQ is the default queueing mode on interfaces that run at E1 speeds (2.048 Mb/s) or below. It is enabled by default for physical interfaces that do not use Link Access Procedure, Balanced (LAPB), X.25, or Synchronous Data Link Control (SDLC) encapsulations. WFQ is not an option for these protocols. WFQ is also enabled by default on interfaces configured for Multilink PPP (MLP). However, if custom queueing (CQ) or priority queueing (PQ) is enabled for a qualifying link, it overrides fair queueing, effectively disabling it. Additionally, WFQ is automatically disabled if you enable autonomous or silicon switching.

The **fair-queue** command is used to enable WFQ on interfaces where it is not enabled by default or was previously disabled.

The syntax for the **fair-queue** command is as follows:

```
fair-queue [congestive-discard-threshold [dynamic-queues  
[reservable-queues]]]
```

Syntax Description

Parameter	Description
<i>congestive-discard-threshold</i>	(Optional) Number of messages allowed in the WFQ system. The default is 64 messages, and a new threshold must be a power of 2 in the range from 16 to 4096. When a conversation reaches this threshold, new message packets are discarded.
<i>dynamic-queues</i>	(Optional) Number of dynamic queues used for best-effort conversations. Values are 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096.
<i>reservable-queues</i>	(Optional) Number of reservable queues used for reserved conversations in the range 0 to 1000. The default is 0. Reservable queues are used for interfaces configured for features such as RSVP.

The syntax for the **fair-queue** command is the same as the syntax for the **fair-queue** command in Cisco IOS Software Release 12.4.

Additional WFQ Configuration Parameters

- Specifies the maximum number of packets that can be in all output queues on the interface at any time.
- The default value for WFQ is 1000.
- Under special circumstances, WFQ can consume a lot of buffers, which may require lowering this limit.

```
router(config-if) #  
hold-queue max-limit out
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6-21

The same **hold-queue** command that can be used with FIFO queuing can also be used with WFQ. The default hold-queue limit with WFQ is 1000 packets.

The WFQ system will generally never reach the hold-queue limit because the CDT limit starts dropping packets of aggressive flows. Under special circumstances, it would be possible to fill the WFQ system. For example, a denial-of-service attack that floods the interface with a large number of packets (each different) could fill all queues at the same rate.

WFQ Configuration Defaults

- Fair queuing is enabled by default:
 - On physical interfaces whose bandwidth is less than or equal to 2.048 Mb/s
 - On interfaces configured for Multilink PPP
- Fair queuing is disabled:
 - If you enable the autonomous or silicon switching engine mechanisms
 - For any sequenced encapsulation: X.25, SDLC, LAPB, reliable PPP

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3—5-22

The figure explains the default behavior of WFQ. WFQ is automatically enabled on all interfaces slower than 2 Mb/s. WFQ is also required on interfaces using Multilink PPP (MLP).

WFQ cannot be used if the switching path does not support WFQ, or if reordering of frames is not allowed because of sequence numbering of Layer 2 frames. WFQ is disabled for any sequenced encapsulation, such as X.25, Synchronous Data Link Control (SDLC), Link Access Procedure, Balanced (LAPB), and reliable PPP.

Monitoring WFQ

This topic describes the Cisco IOS commands required to monitor WFQ on a Cisco router.

Monitoring WFQ

```
router>  
show interface interface
```

- Displays interface delays including the activated queuing mechanism with the summary information

```
Router>show interface serial 1/0  
Hardware is M4T  
Internet address is 20.0.0.1/8  
MTU 1500 bytes, BW 19 Kbit, DLY 20000 usec, rely 255/255, load 147/255  
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 never  
Input queue: 0/75/0 (size/max/drops); Total output drops: 0  
Queueing strategy: weighted fair  
Output queue: 0/1000/64/0 (size/max total/threshold/drops)  
Conversations 0/4/256 (active/max active/max total)  
Reserved Conversations 0/0 (allocated/max allocated)  
5 minute input rate 18000 bits/sec, 8 packets/sec  
5 minute output rate 11000 bits/sec, 9 packets/sec  
... rest deleted ...
```

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-6.23

Use the same **show** commands as with other queuing mechanisms:

- **show interface**
- **show queue**
- **show queueing**

Use the **show interface** command to determine the queuing strategy. The summary statistics are also displayed.

The sample output in the figure shows that there are currently no packets in the WFQ system. The system allows up to 1000 packets (hold-queue limit) with CDT 64. WFQ is using 256 queues. The maximum number of concurrent conversations (active queues) is four.

Monitoring WFQ (Cont.)

router>

```
show queue interface-name interface-number
```

- Displays detailed information about the WFQ system of the selected interface

```
router> show queue serial 1/0
Input queue: 0/75/0 (size/max/drops); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 2/1000/64/0 (size/max total/threshold/drops)
Conversations 2/4/256 (active/max active/max total)
Reserved Conversations 0/0 (allocated/max allocated)

(depth/weight/discards/tail drops/interleaves) 1/4096/0/0/0
Conversation 124, linktype: ip, length: 580
source: 193.77.3.244, destination: 20.0.0.2, id: 0x0166, ttl: 254,
TOS: 0 prot: 6, source port 23, destination port 11033

(depth/weight/discards/tail drops/interleaves) 1/4096/0/0/0
Conversation 127, linktype: ip, length: 585
source: 193.77.4.111 destination: 40.0.0.2, id: 0x020D, ttl: 252,
TOS: 0 prot: 6, source port 23, destination port 11013
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3-5-24

The **show queue** command is used to display the contents of packets inside a queue for a particular interface, including flow (conversation) statistics, as follows:

- Queue depth is the number of packets in the queue.
- Weight is $4096 / (\text{IP Precedence} + 1)$, or $32,384 / (\text{IP Precedence} + 1)$, depending on the Cisco IOS Software version.
- In the command output, discards are used to represent the number of drops due to the CDT limit.
- In the command output, tail drops are used to represent the number of drops due to the hold-queue limit.

The syntax for the **show queue** command in Cisco IOS Software Release 12.4 is as follows:

```
show queue interface-name interface-number [queue-number] [vc
[vp]/] vci]
```

Syntax Description

Parameter	Description
<i>interface-name</i>	The name of the interface.
<i>interface-number</i>	The number of the interface.
<i>queue-number</i>	(Optional) The number of the queue. The queue number is a number from 1 to 16.
<i>vc</i>	(Optional) For ATM interfaces only, shows the fair queueing configuration for a specified permanent virtual circuit (PVC). The name can be up to 16 characters long.
<i>vpi/</i>	(Optional) ATM network virtual path identifier (VPI) for this PVC. The absence of the "/" and a vpi value defaults the vpi value to 0. On the Cisco 7200 and Cisco 7500 Series routers, this value ranges from 0 to 255. The vpi and vci arguments cannot both be set to 0; if one is 0, the other cannot be 0. If this value is omitted, information for all VCs on the specified ATM interface or subinterface is displayed.
<i>vci</i>	(Optional) ATM network virtual channel identifier (VCI) for this PVC. This value ranges from 0 to 1 less than the maximum value set for this interface by the atm vc-per-vp command. Typically, lower values 0 to 31 are reserved for specific traffic [F4 Operation, Administration, and Maintenance (OAM), switched virtual circuit (SVC) signaling, Integrated Local Management Interface (ILMI), and so on] and should not be used. The VCI is a 16-bit field in the header of the ATM cell. The VCI value is unique only on a single link, not throughout the ATM network, because it has local significance only. The vpi and vci arguments cannot both be set to 0; if one is 0, the other cannot be 0.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- The software FIFO queue is basically an extension of the hardware FIFO queue.
- WFQ was developed to overcome the limitations of the more basic queuing methods. Traffic is sorted into **flows** and transmitted by the order of the last bit crossing its channel.
- WFQ classification uses parameters such as source and destination IP addresses, source and destination TCP or UDP ports, transport protocol, and ToS field.
- With WFQ, the CDT is used to start dropping packets of the most aggressive flow, even before the hold-queue limit is reached, and the hold-queue out limit defines the total maximum number of packets that can be in the WFQ system at any time.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-35

Summary (Cont.)

- When determining which packet to transmit next, the WFQ scheduler selects the packet with the lowest finish time; the WFQ implementation in Cisco routers uses the following information to calculate the finish time:
 - Packet length
 - IP precedence number (translated to a "weight" value)
- WFQ benefits: Simple configuration, drops packets of the most aggressive flows. WFQ drawbacks: Not always possible to have one flow per queue, does not allow manual classification, and cannot provide fixed guarantees.
- WFQ is automatically enabled on all interfaces that have a default bandwidth of less than 2 Mb/s. The fair-queue command is used to enable WFQ on interfaces where it is not enabled by default or was previously disabled.
- The same show commands can be used as with other queuing mechanisms: show interface, show queue, and show queuing.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-36

Configuring CBWFQ and LLQ

Overview

Class-based weighted fair queuing (CBWFQ) extends the standard weighted fair queuing (WFQ) functionality to provide support for user-defined traffic classes. With CBWFQ, you define traffic classes based on match criteria including Network-Based Application Recognition (NBAR), access control lists (ACLs), and input interfaces. Packets satisfying the match criteria for a class constitute the traffic for that class. A queue is reserved for each class, and traffic belonging to a class is directed to the queue for that class. Low-latency queuing (LLQ) brings strict priority queuing to CBWFQ. Strict priority queuing allows delay-sensitive data such as voice to be dequeued and sent first (before packets in other queues are dequeued), giving delay-sensitive data preferential treatment over other traffic. This lesson describes how to configure CBWFQ and LLQ.

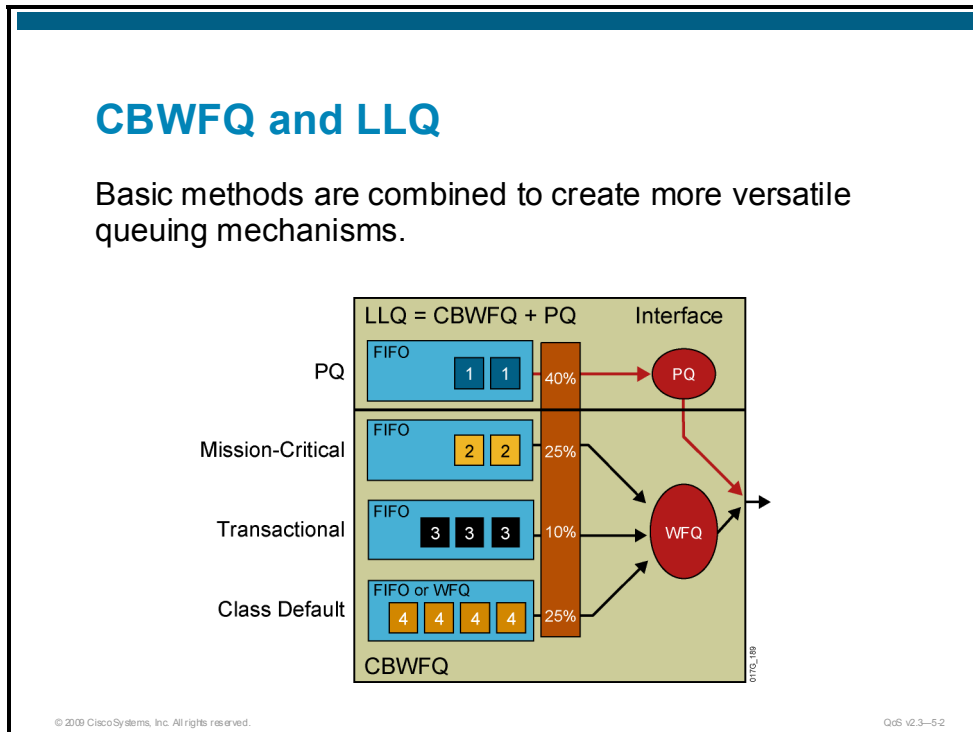
Objectives

Upon completing this lesson, you will be able to configure CBWFQ and LLQ to manage congestion. This ability includes being able to meet these objectives:

- Explain how basic queuing mechanisms can be used to build advanced queuing mechanisms
- Explain the purpose and features of CBWFQ
- Describe CBWFQ features and explain how CBWFQ works using a block diagram
- Describe the benefits of CBWFQ
- Identify the Cisco IOS commands required to configure and monitor CBWFQ on a Cisco router
- Explain the purpose and features of LLQ
- Explain how LLQ works using a block diagram and identify situations in which LLQ is most appropriate for providing QoS
- Describe the benefits of LLQ
- Identify the Cisco IOS commands required to configure and monitor LLQ on a Cisco router

CBWFQ and LLQ

This topic describes how you can use basic queuing mechanisms to build more advanced queuing mechanisms.



Neither the basic queuing methods nor the more advanced WFQ completely solve quality of service (QoS) problems resulting from converged network traffic. These problems remain:

- If only priority queuing (PQ) is used for a voice-enabled network, voice will get the priority needed, but data traffic will likely starve out at some point.
- If only custom queuing (CQ) is used for a voice-enabled network, data traffic will be assured of some bandwidth, but voice traffic will suffer excessive delays.
- If WFQ is used, voice still experiences delay, even when treated “fairly” by WFQ and no bandwidth guarantees can be made for data applications.
- All of the classification, marking, and queuing mechanisms are complicated to use and time-consuming when applied on an interface-by-interface basis.

Newer queuing mechanisms take the best aspects of existing queuing methods and apply them to give voice the priority it requires while still ensuring that data is serviced efficiently on a class basis.

Class-Based Weighted Fair Queuing

This topic describes the purpose and features of class-based weighted fair queuing (CBWFQ).

Class-Based Weighted Fair Queuing

- **CBWFQ** is a mechanism that is used to guarantee bandwidth to classes.
- CBWFQ extends the standard WFQ functionality to provide support for user-defined traffic classes.
 - Classes are based on user-defined match criteria.
 - Packets satisfying the match criteria for a class constitute the traffic for that class.
- A queue is reserved for each class, and traffic belonging to a class is directed to that class queue.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.3

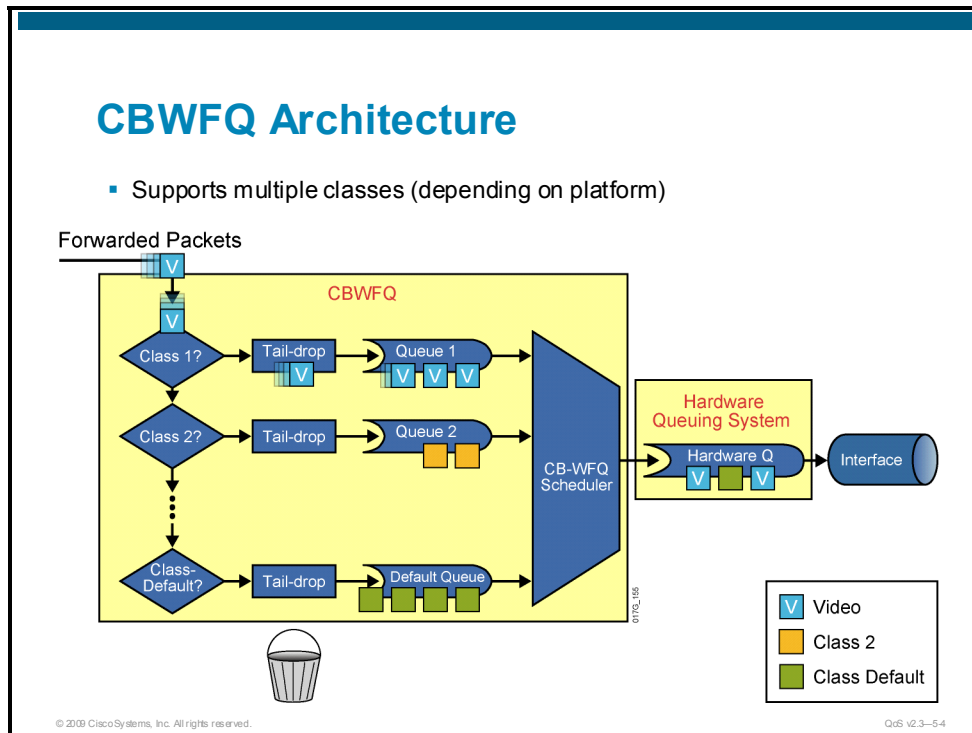
CBWFQ extends the standard WFQ functionality to provide support for user-defined traffic classes. With CBWFQ, you define the traffic classes based on match criteria, including protocols, ACLs, and input interfaces. Packets satisfying the match criteria for a class constitute the traffic for that class. A queue is reserved for each class, and traffic belonging to a class is directed to that class queue.

After a class has been defined according to its match criteria, you can assign characteristics to it. To characterize a class, you assign it bandwidth, weight, and maximum packet limit. The bandwidth assigned to a class is the minimum bandwidth delivered to the class during congestion.

To characterize a class, you also specify the queue limit for that class, which is the maximum number of packets allowed to accumulate in the class queue. Packets belonging to a class are subject to the bandwidth and queue limits that characterize the class. After a queue has reached its configured queue limit, enqueueing of additional packets to the class causes tail drop or random packet drop to take effect, depending on how the class policy is configured.

CBWFQ Architecture

This topic describes the features of CBWFQ and how CBWFQ works.



CBWFQ supports multiple class maps (the number depends upon the platform) to classify traffic into its corresponding FIFO queues. Tail drop is the default dropping scheme of CBWFQ. You can use weighted random early detection (WRED) in combination with CBWFQ to prevent congestion of a class.

The CBWFQ scheduler is used to guarantee bandwidth that is based on the configured weights.

Note Currently, except for the Cisco 7500 Series router platform, all traffic classes (default traffic class excluded) only support FIFO queuing within the class. On all platforms, the default traffic class can support either FIFO or WFQ within the class. Check Cisco.com for the latest information on WFQ support within each traffic class.

CBWFQ Architecture: Classification

- Classification uses **class maps**.
- Availability of certain classification options depends on the Cisco IOS Software version.
- Some classification options depend on the type of interface and encapsulation where the service policy is used.
- For example:
 - Matching on Frame Relay discard eligible bits can be used only on interfaces with Frame Relay encapsulation.
 - Matching on MPLS experimental bits has no effect if MPLS is not enabled.
 - Matching on ISL priority bits has no effect if ISL is not used.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.5

You can use any classification option depending on its availability in the Cisco IOS Software version and the support on the selected interface and encapsulation. These examples illustrate some of the limitations regarding classification options:

- Matching on Frame Relay discard eligible bits can only be used on interfaces with Frame Relay encapsulation.
- Matching on Multiprotocol Label Switching (MPLS) Experimental bits has no effect if MPLS is not enabled.
- Matching on Inter-Switch Link (ISL) priority bits has no effect if ISL is not used.

It is important to note that CBWFQ is configured using Modular QoS command-line interface (CLI), or MQC.

CBWFQ Architecture: Insertion Policy

- Each queue has a maximum number of packets that it can hold (queue size).
- The maximum queue size is platform dependent.
- After a packet is classified to one of the queues, the router will enqueue the packet if the queue limit has not been reached (tail drop within each class).
- WRED can be used in combination with CBWFQ to prevent congestion of the class.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.6

CBWFQ reserves multiple FIFO queues in the WFQ system. The default queue limit is 64 (tail drop) and can be configured with WRED (weighted random drop).

CBWFQ Architecture: Scheduling

- CBWFQ guarantees bandwidth according to weights assigned to traffic classes.
- Weights can be defined by specifying:
 - Bandwidth (in kb/s)
 - Percentage of **bandwidth** (percentage of available interface bandwidth)
 - Percentage of **remaining available bandwidth**
- One service policy can not have mixed types of weights.
- The show interface command can be used to display the available bandwidth.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.7

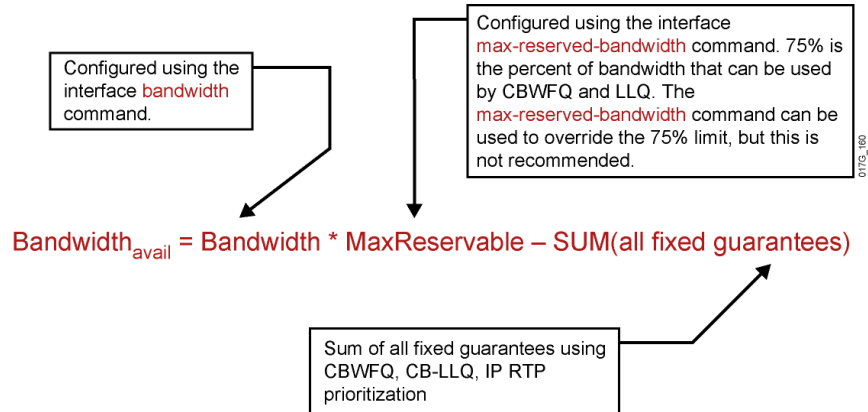
You can configure bandwidth guarantees by using one of these commands:

- The **bandwidth** command allocates a fixed amount of bandwidth by specifying the amount in kb/s. The reserved bandwidth is subtracted from the available bandwidth of the interface where the service policy is used. The allocated bandwidth must also be within the configured reservable limit (75 percent by default).
- You can use the **bandwidth percent** command to allocate a percentage of the default or available bandwidth of an interface. The default bandwidth usually equals the maximum speed of an interface. Sometimes it actually reflects the real speed of an interface (for example, Ethernet or Fast Ethernet). The default value can be replaced by using the **bandwidth** interface command. It is recommended that the bandwidth reflect the real speed of the link. The allocated bandwidth is subtracted from the available bandwidth of the interface in which the service policy is used.
- You can use the **bandwidth remaining percent** command to allocate a portion of the unallocated bandwidth. The bandwidth is *not* subtracted from the available bandwidth of the interface where the service policy is used.

A single service policy cannot mix the fixed bandwidth (in kb/s) and bandwidth percent commands (except with strict priority queues).

CBWFQ Architecture: Available Bandwidth

- Available bandwidth is calculated according to the following formula:



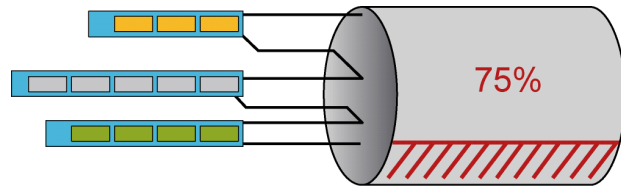
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.8

The available bandwidth displayed by the **show interface** command is calculated by subtracting all fixed bandwidth reservations from 75 percent of the configured bandwidth of an interface.

CBWFQ Architecture: 75 Percent Rule

- Add the following:
 - Class bandwidths
 - RSVP maximum reserved bandwidth
- The result must be less than or equal to 75% of interface bandwidth (or Frame Relay, DLCI, CIR).
 - This leaves headroom for overhead traffic such as Layer 2 keepalives and bandwidth for the class default traffic.
- The 75% rule is conservative.
- The **max-reserved-bandwidth** command overrides the 75% limit but is seldom recommended.



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.9

Properly provisioning the network bandwidth is a major component of successful network design. You can calculate the required bandwidth by adding the bandwidth requirements for each major application (for example, voice, video, and data). The resulting sum represents the minimum bandwidth requirement for any given link, and it should not exceed 75 percent of the total available bandwidth for the link (or Frame Relay or ATM PVC committed information Rate [CIR]). This 75-percent rule assumes that some bandwidth is required for overhead traffic, such as layer 2 keepalive messages, as well as for the class default traffic.

The total amount of bandwidth allocated for all classes included in a policy map should not exceed 75 percent of the available bandwidth on the interface. The **max-reserved bandwidth** command overrides the 75 percent limitation, but overriding is recommended only for the most knowledgeable network administrators who have access to precise figures for available, used, and required bandwidth, including control plane traffic bandwidth requirements. If all of the bandwidth is not allocated, the remaining bandwidth is proportionally allocated among the classes based on the configured bandwidth of the classes.

CBWFQ Benefits

This topic describes the benefits of CBWFQ.

CBWFQ Benefits

■ Benefits

- Minimum bandwidth allocation
- Finer granularity and scalability
- MQC interface easy to use
- Maximizes transport of priority traffic
- Weights guarantee minimum bandwidth
- Unused capacity shared among the other classes
- Queues separately configured for QoS

■ Drawbacks

- Voice traffic can still suffer unacceptable delay

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6-10

CBWFQ allows you to define traffic classes based on custom-defined match criteria such as ACLs, input interfaces, protocol, and QoS label. For example, a class might consist of a team working on a certain project, or a class can be created for the important mission-critical applications such as enterprise resource planning (ERP) applications. When the traffic classes have been defined, they can be assigned a bandwidth, queue limit, or drop policy such as WRED. These are additional benefits of CBWFQ:

- **Bandwidth allocation:** CBWFQ allows you to specify the exact amount of bandwidth to be allocated for a specific class of traffic. Accounting for available bandwidth on the interface, you can configure multiple classes (the number depends upon the platform).
- **Finer granularity and scalability:** CBWFQ allows you total flexibility to define a class, based on ACLs and protocols or input interfaces, thereby providing finer granularity.
- **Supported by MQC:** CBWFQ is supported by the easy-to-use MQC.

The CBWFQ feature is supported on all platforms that WFQ is supported on; in other words, the Cisco 7200, 3800, 1800, and 800 Series routers, and others.

Configuring and Monitoring CBWFQ

This topic describes the Cisco IOS commands that are used to configure and monitor CBWFQ on a Cisco router.

Configuring CBWFQ

```
router(config-pmap-c)#
```

```
bandwidth {bandwidth-kbps | remaining percent percentage |  
percent percentage}
```

- The **bandwidth** command allocates a fixed amount of bandwidth to a class.
- Use of the **percent** keyword enables you to allocate a percentage of bandwidth to a class; the configured (or default) interface bandwidth is used to calculate the guaranteed bandwidth.
- Use of the **remaining percent** keywords enables you to allocate a percentage of available bandwidth to a class.

```
router(config-pmap-c)#
```

```
queue-limit queue-limit
```

- The **queue-limit** command sets the maximum number of packets the queue can hold; the default maximum is 64.

© 2009 Cisco Systems, Inc. All rights reserved.QoS 42.3-6-11

The **bandwidth** policy-map class configuration command is used to specify or modify the bandwidth allocated for a class belonging to a policy map.

All classes belonging to one policy map should use the same type of fixed bandwidth guarantee, in kb/s, percentage of interface bandwidth, or percentage of available bandwidth.

Configuring bandwidth in percentages is most useful when the underlying link bandwidth is unknown or the relative class bandwidth distributions are known.

The syntax for the **bandwidth** command is as follows:

```
bandwidth {bandwidth-kbps | remaining percent percentage |  
percent percentage}
```

Syntax Description

Parameter	Description
<i>bandwidth-kbps</i>	Amount of bandwidth, in kb/s, to be assigned to the class.
remaining percent <i>percentage</i>	Amount of guaranteed bandwidth, based on a relative percent of available bandwidth. The percentage can be a number from 1 to 100.
percent <i>percentage</i>	Amount of guaranteed bandwidth, based on an absolute percent of available bandwidth. The percentage can be a number from 1 to 100. (By default, only 75 percent can be reserved.)

Note The syntax shown here for the **bandwidth** command is the same as the syntax for the **bandwidth** command in Cisco IOS Software Release 12.4.

These restrictions apply to the **bandwidth** command:

- If the **percent** keyword is used, the sum of the class bandwidth percentages cannot exceed 100 percent.
- The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.
- A policy map can have all the class bandwidths specified in kb/s or in percentages, but not a mix of both. However, the unit for the **priority** command in the priority class can be different from the bandwidth unit of the low-priority class. CBWFQ is not supported on sub-interfaces. If a service policy is applied to a sub-interface in which the service policy directly references a policy map that has a bandwidth statement in it, this error message will be displayed:

```
router(config-subif)#service-policy output cbwfq
CBWFQ : NOT supported on subinterfaces
```

You can change the default queue limit of 64 packets by using the **queue-limit** command. It is recommended that you not change the default value.

The syntax for the **queue-limit** command in Cisco IOS Software Release 12.4 is as follows:

```
queue-limit queue-limit-size [bytes | ms | packets]
```

Syntax Description

Parameter	Description
<i>queue-limit-size</i>	The maximum size of the queue. The maximum varies according to the optional unit of measure keyword specified (bytes , ms , or packets). If an optional unit of measure is not indicated, the default unit of measure is packets.
bytes	(Optional) Indicates that the unit of measure is bytes. Valid range for bytes is a number from 1 to 8192000. The bytes keyword is not supported on Cisco 7600 and ASR 1000 series routers.
ms	(Optional) Indicates that the unit of measure is milliseconds. Valid range for milliseconds is a number from 1 to 3400. The ms keyword is not supported on Cisco 7600 and ASR 1000 series routers.
packets	(Optional) Indicates that the unit of measure is packets. Valid range for packets is a number from 1 to 32768 but can also vary by platform and release.

Configuring CBWFQ (Cont.)

```
router(config)#access-list 101 permit udp host 10.10.10.10
host 10.10.10.20 range 16384 20000
router(config)#access-list 102 permit udp host 10.10.10.10
host 10.10.10.20 range 53000 56000
router(config)# class-map class1
router(config-cmap)# match access-group 101
router(config-cmap)# exit
router(config)# class-map class2
router(config-cmap)# match access-group 102
router(config-cmap)# exit

router(config)# policy-map policy1
router(config-pmap)# class class1
router(config-pmap-c)# bandwidth 3000
router(config-pmap-c)# queue-limit 30
router(config-pmap-c)# exit
router(config-pmap)# class class2
router(config-pmap-c)# bandwidth 2000
router(config-pmap-c)# exit
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--5-12

The sample configuration shows how CBWFQ is used to guarantee bandwidth to each of the two classes.

Configuring the Class-Default Class Policy

```
policy-map POLICY8
class class-default
  bandwidth 1000
  queue-limit 40
```

- FIFO queuing is configured within the default class.
- The default class is guaranteed 1 Mb/s of bandwidth and the maximum queue size is limited to 40 packets.

```
policy-map POLICY9
class class-default
  fair-queue 16
  queue-limit 20
```

- Sixteen dynamic queues are reserved for use by flow-based WFQ running on the default class.
- Because the **queue-limit** command is configured, tail drop is used for each dynamic queue when the maximum number of packets are enqueued and additional packets arrive.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-13

The class-default class is used to classify traffic that does not fall into one of the defined classes. Once a packet is classified, all of the standard mechanisms that can be used to differentiate service among the classes apply. The class-default class is predefined when you create a policy map, but you must configure it. If no default class is configured, then by default, the traffic that does not match any of the configured classes is flow classified and given best-effort treatment.

By default, the class-default class is defined as flow-based WFQ. However, configuring the default class with the **bandwidth policy-map class** configuration command disqualifies the default class as flow-based WFQ.

To configure a policy map and configure the class-default class to use tail drop, use the **policy-map** command in global configuration mode to specify the policy map name; then configure policy for the default class by using the **bandwidth command** (or the **fair-queue** command for class-default) and the **queue-limit** command in policy-map class configuration mode.

If you want the class-default class to use WRED packet drop instead of tail drop, you can also use the **bandwidth command** (or the **fair-queue** command for class-default) with the **random-detect** and **random-detect exponential-weighting-constant** (or **random-detect precedence**) commands in policy-map class configuration mode to configure the policy for the default class.

The **fair-queue** command can be used to specify the number of dynamic queues to be reserved for use by flow-based WFQ running on the default class. The number of dynamic queues is derived from the bandwidth of the interface. Refer to the tables accompanying the description of the fair-queue (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface or ATM VC.

Monitoring CBWFQ

```
router>
```

```
show policy-map interface [interface]
```

- Displays parameters and statistics of CBWFQ

```
router>show policy-map interface  
FastEthernet0/0
```

```
Service-policy output: Policy1
```

```
Class-map: Class1 (match-any)
```

```
0 packets, 0 bytes
```

```
5 minute offered rate 0 bps, drop rate 0 bps
```

```
Match: any
```

```
Weighted Fair Queueing
```

```
Output Queue: Conversation 265
```

```
Bandwidth remaining 20 (%) 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)
```

```
42 packets, 4439 bytes
```

```
5 minute offered rate 0 bps, drop rate 0 bps
```

```
Match: any
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3--5-14

The **show policy-map interface** command displays all service policies applied to the interface. The output displays the policing parameters and statistics.

The syntax shown in the figure for the **show policy-map interface** command is compatible with the syntax for the **show policy-map interface** command in Cisco IOS Software Release 12.4.

Low-Latency Queuing

This topic describes the purpose and features of low-latency queuing (LLQ).

Low-Latency Queuing

- Priority queue added to CBWFQ for real-time traffic.
- High-priority classes are guaranteed:
 - Low-latency propagation of packets
 - Bandwidth
- High-priority classes are also policed when congestion occurs; then they cannot exceed their guaranteed bandwidth.
- Lower-priority classes use CBWFQ.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6, 15

While WFQ provides a fair share of bandwidth to every flow, and provides fair scheduling of its queues, it cannot provide guaranteed bandwidth and low delay to select applications. For example, voice traffic may still compete with other aggressive flows in the WFQ queuing system because the WFQ system lacks priority scheduling for time-critical traffic classes.

The LLQ feature brings strict priority queuing to CBWFQ. Strict priority queuing allows delay-sensitive data such as voice to be dequeued and sent first (before packets in other queues are dequeued), giving delay-sensitive data preferential treatment over other traffic.

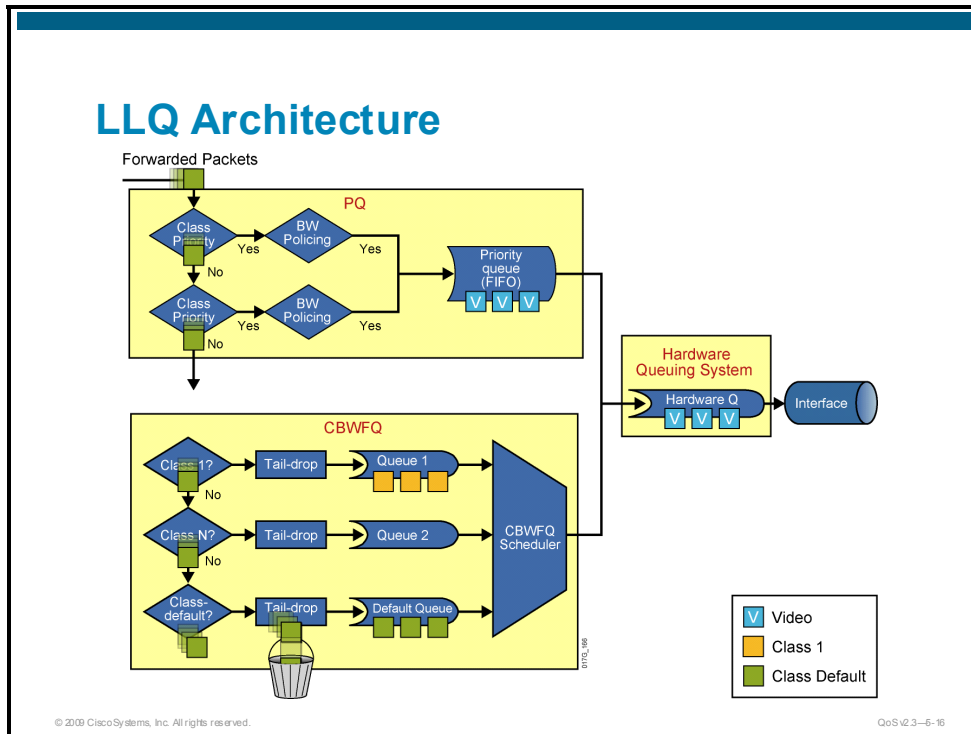
Without LLQ, CBWFQ provides weighted fair queuing based on defined classes with no strict priority queue available for real-time traffic. CBWFQ allows you to define traffic classes and then assign characteristics to that class. For example, you can designate the minimum bandwidth delivered to the class during congestion.

For CBWFQ, the weight for a packet belonging to a specific class is derived from the bandwidth that you assigned to the class when you configured it. Therefore, the bandwidth assigned to the packets of a class determines the order in which packets are sent. All packets are serviced fairly based on weight; no class of packets may be granted strict priority. This scheme poses problems for voice traffic, which is largely intolerant of delay, especially variation in delay. For voice traffic, variations in delay introduce irregularities of transmission that are heard as jitter in the conversation.

The LLQ feature provides strict priority queuing for CBWFQ, reducing jitter in voice conversations. Configured by the **priority** command, LLQ enables use of a single, strict priority queue within CBWFQ at the class level, allowing you to direct traffic belonging to a class to the CBWFQ strict priority queue. To enqueue class traffic to the strict priority queue, you configure the **priority** command for the class after you specify the named class within a policy map. Classes to which the priority command is applied are considered priority classes. Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

LLQ Architecture

This topic describes how LLQ works and identifies situations in which LLQ is most appropriate for providing QoS.



When CBWFQ is configured as the queuing system, it creates a number of queues, into which it classifies traffic classes. These queues are then scheduled with a WFQ-like scheduler, which can guarantee bandwidth to each class.

If LLQ is used within the CBWFQ system, it creates an additional priority queue in the WFQ system, which is serviced by a strict priority scheduler. Any class of traffic can therefore be attached to a service policy, which uses priority scheduling, and hence can be prioritized over other classes.

LLQ Benefits

This topic describes the benefits of LLQ.

LLQ Benefits

- High-priority classes are guaranteed:
 - Low-latency propagation of packets
 - Bandwidth
- Consistent configuration and operation across all media types.
- Entrance criteria to a class can be defined by an ACL:
 - Not limited to UDP ports as with IP RTP priority
 - Defines trust boundary to ensure simple classification and entry to a queue

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--5-17

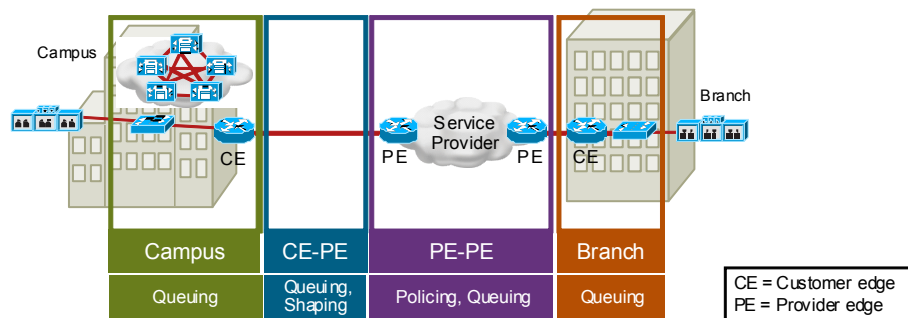
The LLQ priority scheduler guarantees both low-latency propagation of packets and bandwidth to high-priority classes. Low latency is achieved by expediting traffic using a priority scheduler. Bandwidth is also guaranteed by the nature of priority scheduling, but is policed to a user-configurable value. The strict PQ scheme allows delay-sensitive data such as voice to be dequeued and sent first—that is, before packets in other queues are dequeued. Delay-sensitive data is given preferential treatment over other traffic.

The LLQ feature provides strict PQ on ATM virtual circuits (VCs); the IP Real-Time Transport Protocol (RTP) priority feature only allows PQ on interfaces. Because you can configure the priority status for a class within CBWFQ, you are not limited to User Datagram Protocol (UDP) port numbers to stipulate priority flows (which were necessary with IP RTP). Instead, all of the valid match criteria used to specify traffic for a class now applies to priority traffic.

Policing of priority queues also prevents the priority scheduler from monopolizing the CBWFQ scheduler and starving nonpriority classes, as legacy PQ does. By configuring the maximum amount of bandwidth allocated for packets belonging to a class, you can avoid starving nonpriority traffic.

LLQ and Cisco TelePresence

- Queuing must be enabled on every node along the path.
- It is recommended to place Cisco TelePresence in a strict-priority queue.
- Service level requirements for TelePresence are as follows:
 - Latency: 150 ms
 - Jitter: 10 ms
 - Packet loss: 0.05%



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-18

To achieve the high levels of service required by the Cisco TelePresence experience, queuing must be enabled on every node along the path to provide service guarantees, regardless of how infrequently congestion may occur on certain nodes. (Congestion can and does occur even on very high-bandwidth mediums). If queuing is not properly configured on every node, the Cisco TelePresence experience cannot be guaranteed.

RFC 4594 specifies the minimum queuing requirement of the Real-Time Interactive service class to be a rate-based queue (a queue that has a guaranteed minimum bandwidth rate). However, RFC 4594 makes an allowance that while the PHB for the Real-Time Interactive service class should be configured to provide high bandwidth assurance, it may be configured as a second EF PHB that uses relaxed performance parameters, a rate scheduler, and a CS4 DSCP value.

This means that Cisco TelePresence, which has been assigned to this Real-Time Interactive service class, can be queued with either a guaranteed rate nonpriority queue such as CBWFQ or a guaranteed-rate strict priority queue such as LLQ; in either case, Cisco TelePresence is to be marked as Class Selector 4 (and not EF).

Therefore, since RFC 4594 allows for the real-time interactive service-class to be given a second EF PHB and because of the low latency, low jitter, and low loss requirements of Cisco TelePresence, it is recommended to place Cisco TelePresence in a strict-priority queue, such as an LLQ queue or a Cisco Catalyst hardware priority queue, whenever possible.

An additional provisioning consideration must be taken into account when provisioning Cisco TelePresence with a second EF PHB, which relates to the amount of bandwidth of a given link that should be assigned for strict priority queuing. The well-established and widely deployed Cisco best-practice recommendation is to limit the amount of strict priority queuing configured on an interface to no more than one-third of the capacity of the link. This is commonly referred to as “the 33 percent LLQ rule.”

The rationale behind this rule is that if you assign too much traffic for strict priority queuing, the overall effect is a dampening of QoS functionality for non-real-time applications. The goal of convergence is to enable voice, video, and data to transparently co-exist on a single network. When real-time applications such as voice and Cisco TelePresence dominate a link, data applications fluctuate significantly in their response times if Cisco TelePresence calls are present versus when they are absent, thus destroying the transparency of the converged network.

Note	Cisco IOS Software allows the configuration of multiple strict-priority LLQ queues. In such a multiple-LLQ context, this design principle means that the sum of all LLQ queues should be within one-third of link capacity.

The 33-percent LLQ rule is a conservative and safe design ratio for merging real-time applications with data applications. This strict priority queuing rule is simply a best-practice design recommendation and is not a mandate. There may be cases where specific business objectives cannot be met while holding to this recommendation. In such cases, enterprises must provision according to their detailed requirements and constraints. However, it is important to recognize the tradeoffs involved with over-provisioning strict priority traffic and its negative performance impact on non-real-time-application response times.

It is also worth noting that the 33 percent rule applies only for converged networks. If you choose to deploy dedicated WAN circuits for Cisco TelePresence traffic, the 33 percent rule does not apply. This is because Cisco TelePresence (and perhaps some nominal amount of management and signaling traffic) is the only traffic on the circuit. In these cases, you can use up to 98 percent of the link capacity for Cisco TelePresence and reserve 2 percent for routing protocols, network management traffic such as SSH and SNMP, and signaling.

Note	For more information on QoS and Cisco TelePresence, refer to the <i>Cisco TelePresence Network Systems 2.0 Design Guide</i> on Cisco.com.

Configuring and Monitoring LLQ

This topic describes the Cisco IOS commands that are used to configure and monitor LLQ on a Cisco router.

Configuring LLQ

```
router(config-pmap-c) #
```

```
priority bandwidth [burst]
```

- Allocates a fixed amount of bandwidth (in kb/s) to a class and ensures expedited forwarding.
- Traffic exceeding the specified bandwidth is dropped if congestion exists; otherwise, policing is not used.

```
router(config-pmap-c) #
```

```
priority percent percentage [burst]
```

- Allocates a percentage of configured or default interface bandwidth to a class and ensures expedited forwarding.
- Traffic exceeding the specified bandwidth is dropped if congestion exists; otherwise, policy is not used.

© 2003 Cisco Systems, Inc. All rights reserved.QoS v2.3-6-10

When you specify the **priority** command for a class, it takes a bandwidth argument that gives maximum bandwidth in kb/s. You use this parameter to specify the maximum amount of bandwidth allocated for packets belonging to the class configured with the **priority** command. The bandwidth parameter both guarantees bandwidth to the priority class and restrains the flow of packets from the priority class. The syntax for the **priority** command is as follows:

```
priority {bandwidth-kbps | percent percentage} [burst]
```

Parameter	Description
<i>bandwidth-kbps</i>	Guaranteed allowed bandwidth, in kb/s, for the priority traffic. The amount of guaranteed bandwidth varies according to the interface and platform in use. Beyond the guaranteed bandwidth, the priority traffic will be dropped in the event of congestion to ensure that the nonpriority traffic is not starved.
percent	Specifies that the amount of guaranteed bandwidth will be specified by the percent of available bandwidth.
<i>percentage</i>	Used in conjunction with the percent keyword; specifies the percentage of the total available bandwidth to be set aside for the priority class. The percentage can be a number from 1 to 100. (By default, only 75 percent can be reserved.)
burst	(Optional) Specifies the burst size, in bytes. The range of the burst is 32 to 2,000,000 bytes. The burst size allows temporary bursting of traffic above the maximum limit after a period of inactivity. The default burst value is computed as 200 ms of traffic at the configured bandwidth rate.

Note The syntax shown here for the **priority** command is the same as the syntax for the **priority** command in Cisco IOS Software Release 12.4.

In the event of congestion, if the priority class traffic exceeds the bandwidth guarantee, a congestion-aware policer is used to drop the exceeding packets. Voice traffic enqueued to the priority queue is UDP-based and therefore not adaptive to the early packet-drop characteristic of WRED. Because WRED is ineffective, you cannot use the WRED **random-detect** command with the **priority** command. In addition, because policing is used to drop packets and a queue limit is not imposed, the **queue-limit** command cannot be used with the **priority** command.

When congestion occurs, traffic destined for the priority queue is metered to ensure that the bandwidth allocation configured for the class to which the traffic belongs is not exceeded.

Priority traffic metering has these qualities:

- It is much like committed access rate (CAR), except that priority traffic metering is performed only under conditions of congestion. When the device is not congested, the priority-class traffic is allowed to exceed its allocated bandwidth. When the device is congested, the priority-class traffic above the allocated bandwidth is discarded.
- Metering performed on a per-packet basis, and tokens, are replenished as packets are sent. If not enough tokens are available to send the packet, the packet is dropped.
- It restrains priority traffic to its allocated bandwidth to ensure that nonpriority traffic, such as routing packets and other data, is not starved.

With metering, the classes are policed and rate-limited individually. That is, although a single policy map might contain four priority classes, all of which are enqueued in a single priority queue, they are each treated as separate flows with separate bandwidth allocations and constraints.

Note It is important to note that because bandwidth for the priority class is specified as a parameter to the **priority** command, you cannot also configure the **bandwidth** command for a priority class. To do so is a configuration violation that would only introduce confusion in relation to the amount of bandwidth to allocate.

Keep these guidelines in mind when using the **priority** command:

- You account for Layer 2 encapsulations in the amount of bandwidth that you specify with the **priority** command. However, ensure that you configure a bandwidth with room for the cell-tax overhead.
- Use the **priority** command for VoIP on serial links and ATM permanent virtual circuits (PVCs).

Example: Calculating LLQ Bandwidth Required for VoIP

The bandwidth consumed by VoIP streams is calculated by adding the packet payload and all the headers, then multiplying that total number by the per-second packet rate.

The example shows how to calculate the VoIP bearer bandwidth requirement for a single VoIP call using a G.711 codec:

G.711 = 160 bytes payload size

Packet size = payload size + IP/UDP/RTP headers
= 160 bytes + 20 bytes + 8 bytes + 12 bytes
= 200 bytes

Sampling Rate = 20 msec per sample = 50 samples per second

Bandwidth (bytes/sec) without Layer 2 overhead
= 200 bytes/packet x 50 packets/second
= 10000 bytes/second

Bandwidth (bits/sec) without Layer 2 overhead
= 10000 bytes/second * 8 bits/byte
= 80000 bits/second (80 kbps)

Bandwidth (bits/sec) with Layer 2 overhead

= 80000 bits/second + L2 overhead bytes/second

Configuring LLQ (Cont.)

```
class-map voip
  match ip precedence 5
!
class-map mission-critical
  match ip precedence 3 4
!
class-map transactional
  match ip precedence 1 2
!
policy-map Policy1
  class voip
    priority percent 10
  class mission-critical
    bandwidth percent 30
    random-detect
  class transactional
    bandwidth percent 20
    random-detect
  class class-default
    fair-queue
    random-detect
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3—5-20

The figure shows a configuration example in which the VoIP traffic class, classified by the IP Precedence of 5, is queued in a LLQ within the CBWFQ system. The VoIP class receives guaranteed priority scheduling over the other classes but is limited to 10 percent of the bandwidth.

Monitoring LLQ

```
router>
```

```
show policy-map interface interface
```

- Displays the packet statistics of all classes that are configured for all service policies either on the specified interface or subinterface.

```
router>show policy-map interface fastethernet 0/0
FastEthernet0/0

Service-policy output: LLQ

  Class-map: LLQ (match-any)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
    Weighted Fair Queueing
      Strict Priority
        Output Queue: Conversation 264
        Bandwidth 1000 (kbps) Burst 25000 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

  Class-map: class-default (match-any)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-21

The **show policy-map interface** command displays the packet statistics of all classes that are configured for all service policies on the specified interface. Some of the key fields in the command output are described as follows:

Parameter	Description
Class-map	Class of traffic being displayed. Output is displayed for each configured class in the policy.
offered rate	Rate, in kb/s, of packets coming in to the class.
drop rate	Rate, in kb/s, at which packets are dropped from the class. The drop rate is calculated by subtracting the number of successfully transmitted packets from the offered rate.
Match	Match criteria specified for the class of traffic.
pkts matched/bytes matched	Number of packets (also shown in bytes) matching this class that were placed in the queue.
depth/total drops/no-buffer drops	Number of packets discarded for this class. No-buffer indicates that no memory buffer exists to service the packet.

Note Syntax for commands documented in this lesson may vary by platform and software version. For more information on command syntax for your Cisco IOS Software release and router platform, please refer to the *Cisco IOS Master Command List, Release 12.4*, on Cisco.com.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- By combining basic queuing mechanisms, you can build an advanced queuing mechanism.
- CBWFQ is a mechanism that is used to overcome deficiencies of WFQ.
- CBWFQ extends the standard WFQ functionality to provide support for traffic classes. Classes are based on user-defined match criteria.
- CBWFQ provides a minimum bandwidth guarantee according to traffic classes.
- CBWFQ uses the **policy-map** configuration commands to configure such parameters as bandwidth and queue limit within the associated class; the **show policy-map interface** command is used to display the CBWFQ and LLQ statistics.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-32

Summary (Cont.)

- LLQ extends the functionality of CBWFQ by adding a priority queue for time-sensitive traffic such as voice and video; the LLQ scheduler guarantees both low latency and bandwidth for the traffic in the priority queue.
- LLQ is implemented within CBWFQ by the addition of a priority queue that is serviced using a strict priority scheduler. In the event of congestion, if the priority queue traffic exceeds the bandwidth guarantee, a congestion-aware policer is used to drop the exceeding traffic.
- LLQ allows delay-sensitive data such as voice to be given preferential treatment over other traffic.
- LLQ is configured using the **priority** command in policy-map class configuration mode.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-33

Configuring LAN Congestion Management

Overview

In a converged network, it is imperative that you implement quality of service (QoS). You need to be aware of how queuing is implemented across the Cisco Catalyst switching family, because different switches offer varied mechanisms for providing QoS. This lesson describes how several key Catalyst switches provide queuing support for QoS and how to configure those switches for QoS. The lesson also describes priority queuing (PQ), weighted round robin (WRR), shaped round robin (SRR), and weighted tail drop (WTD).

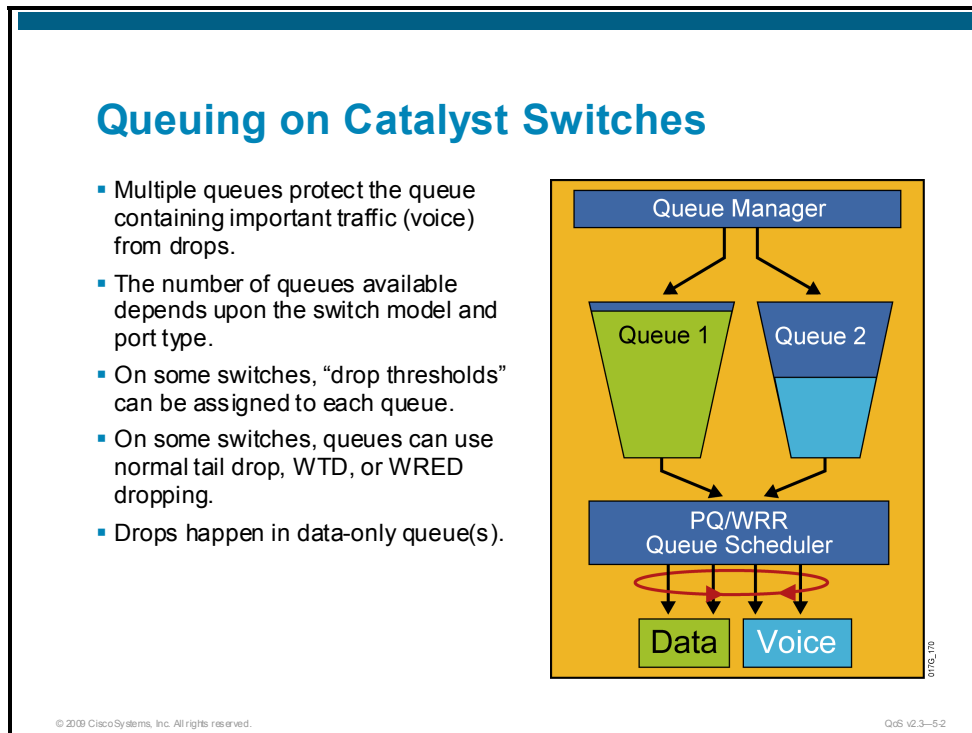
Objectives

Upon completing this lesson, you will be able to configure WRR on a Catalyst switch to manage LAN congestion. This ability includes being able to meet these objectives:

- Describe the different queuing capabilities available on Cisco Catalyst switches
- Explain how WRR works on the Catalyst 2950 switch
- Describe the commands required to configure CoS to queue mappings for PQ on Catalyst 2950 switches
- Describe the commands required to configure WRR on Catalyst 2950 switches
- Explain how WTD works on the Catalyst 2960 switch
- Describe the commands required to configure WTD on Catalyst 2960 switches
- Explain how SRR works on the Catalyst 2960 switch
- Describe the commands required to configure SRR on Catalyst 2960 switches
- Describe the commands required to monitor queuing on Catalyst 2950 and 2960 switches

Queuing on Catalyst Switches

This topic describes the different queuing capabilities available on Cisco Catalyst switches.



In a converged network, it is vital to ensure that voice traffic is not dropped. The use of multiple queues in Catalyst switches protects the queue containing important traffic (voice) from being dropped. Cisco Catalyst switches offer a variety of queuing capabilities, depending upon the switch model and port type.

One of the key options that you can assign to queues in most Catalyst switches is “drop thresholds.” You can assign one or more drop thresholds to a queue. Packets are queued until the thresholds are exceeded.

For example, all packets with differentiated services code points (DSCPs) that are assigned to the first threshold are dropped until the threshold is no longer exceeded. However, packets assigned to a second threshold continue to be queued and sent as long as the second threshold is not exceeded. The thresholds are all specified as percentages ranging from 1 to 100. A value of 10 indicates a threshold in which the buffer is 10 percent full.

On some switches, queues can use normal tail drop, WTD, or weighted random early detection (WRED) dropping.

Drops will occur only in data queues. The purpose of using multiple queues is to prevent voice traffic from being dropped or delayed.

Queuing on Catalyst Switches (Cont.)

- Key queuing features depend upon the switch hardware:
 - Number of queues per port
 - Type of queues (priority or standard)
 - Ability to have drop thresholds for a queue
 - Number of drop thresholds per queue
 - Type of drop thresholds (tail drop or WRED)
- Switch queuing capabilities are shown as:
 - 2Q2T:
 - Two queues
 - Two drop thresholds for each queue
 - 1P2Q2T:
 - One priority queue
 - Two additional queues
 - Two drop thresholds for each queue

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.3

Different Cisco Catalyst switches offer these different queuing capabilities:

- The number of queues per port
- The type of queues (priority or standard)
- The capability to have drop thresholds for a queue
- The number of drop thresholds per queue
- The type of drop thresholds (tail drop or WRED)

When you view information on Cisco Catalyst switches, queuing information is displayed in an abbreviated format, as in these examples:

- **2Q2T:** Indicates that the switch supports two standard queues and two drop thresholds per queue
- **1P2Q2T:** Indicates that the switch supports one priority queue, two standard queues, and two drop thresholds per queue

Queuing on Catalyst Switches (Cont.)

	6500	4500	2960, 2970, 3560, 3750	3550	2950
Transmit Queues	2Q2T 1P2Q2T 1P3Q1T 1P2Q1T	1P3Q1T 4Q1T	4Q3T 1P3Q3T	1P3Q2T 4Q2T	1P3Q 4Q
Receive Queues	1Q4T 1P1Q4T 1P1Q 1P1Q8T	No	1P1Q3T 2Q3T	No	No

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.4

The chart shows the capabilities of Catalyst switch models.

6500 Series Catalyst Switches

The Catalyst 6500 provides both receive (Rx) and transmit (Tx) queues. The number and type of queues is dependent upon the line card.

The Rx queues are designed to protect voice traffic from delays or drops. An example of the implementation of an Rx queue with a priority queue and a standard queue with drop thresholds (1P1Q4T) is as follows:

- Frames with class of service (CoS) 5 go to the priority queue.
- Frames with CoS 0, 1, 2, 3, 4, 6, or 7 go to the standard Rx queue as follows:
 - Using standard receive-queue tail-drop threshold 1, the switch drops incoming frames with CoS 0 or 1 when the receive-queue buffer is 50 percent or more full.
 - Using standard receive-queue tail-drop threshold 2, the switch drops incoming frames with CoS 2 or 3 when the receive-queue buffer is 60 percent or more full.
 - Using standard receive-queue tail-drop threshold 3, the switch drops incoming frames with CoS 4 when the receive-queue buffer is 80 percent or more full.
 - Using standard receive-queue tail-drop threshold 4, the switch drops incoming frames with CoS 6 or 7 when the receive-queue buffer is 100 percent full.

An example of the implementation of a Tx queue with two queues and two drop thresholds (2Q2T) on the Cisco Catalyst 6500 Series switch is as follows. For 2Q2T ports, each Tx queue has two tail-drop thresholds that function as follows:

- Frames with CoS 0, 1, 2, or 3 go to the low-priority Tx queue (queue 1):
 - Using Tx queue 1, tail-drop threshold 1, the switch drops frames with CoS 0 or 1 when the low-priority transmit-queue buffer is 80 percent full.

- Using Tx queue 1, tail-drop threshold 2, the switch drops frames with CoS 2 or 3 when the low-priority transmit-queue buffer is 100 percent full.
- Frames with CoS 4, 5, 6, or 7 go to the high-priority Tx queue (queue 2):
 - Using Tx queue 2, tail-drop threshold 1, the switch drops frames with CoS 4 or 5 when the high-priority transmit-queue buffer is 80 percent full.
 - Using Tx queue 2, tail-drop threshold 2, the switch drops frames with CoS 6 or 7 when the high-priority transmit-queue buffer is 100 percent full.

4500 Series Catalyst Switches

On the Catalyst 4500 series switch with a Supervisor III engine, each physical port has four Tx queues (egress queues). Each packet that needs to be transmitted is enqueued to one of the Tx queues. The Tx queues are then serviced based on the Tx queue scheduling algorithm.

When the final transmit DSCP is computed (including any markdown of DSCP), the transmit DSCP-to-Tx queue mapping configuration determines the Tx queue. The packet is placed in the Tx queue of the transmit port, determined from the transmit DSCP.

The four Tx queues for a transmit port share the available link bandwidth of that transmit port. You can set the link bandwidth to be shared differently among the Tx queues using the **bandwidth** command in interface Tx queue configuration mode. With this command, you assign the minimum guaranteed bandwidth for each Tx queue.

By default, all queues are scheduled in a round-robin manner.

You can configure Tx queue 3 on each port with higher priority. When Tx queue 3 is configured with higher priority, packets in Tx queue 3 are scheduled ahead of packets in other queues.

When Tx queue 3 is configured at a higher priority, the packets are scheduled for transmission before the other Tx queues only if queue 3 has not met the allocated bandwidth sharing configuration—that is, if continuing to give priority to queue 3 would impact the minimum guaranteed to the other queues. Any traffic that exceeds the configured shape rate will be queued and transmitted at the configured rate. If the burst of traffic exceeds the size of the queue, packets will be dropped to maintain transmission at the configured shape rate.

Drop thresholds can be configured as tail drop or Dynamic Buffer Limiting (DBL).

3550 Catalyst Switches

On the Cisco Catalyst 3550 Series switches, the default scheduling method is WRR with 25 percent of bandwidth per queue by default.

You can assign CoS values to queues during configuration. The default CoS-to-queue assignment is as follows:

- CoS 6 to 7 placed in queue 4
- CoS 4 to 5 placed in queue 3
- CoS 2 to 3 placed in queue 2
- CoS 0 to 1 placed in queue 1

The switches support WRR scheduling, and WRR with a priority queue, as follows:

- The WRR scheduling algorithm ensures that lower-priority packets are not entirely starved for bandwidth and are serviced without compromising the priority settings administered by the network manager.
- WRR with a priority queue ensures that higher-priority packets will always get serviced first, ahead of other traffic in lower-priority queues. The priority queue is defined as queue 4.

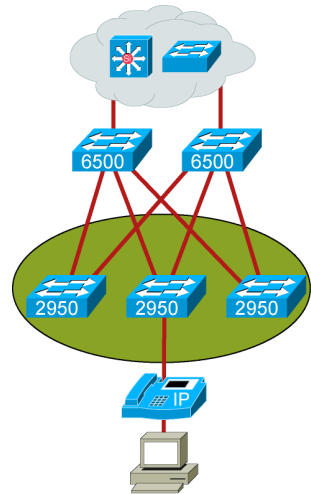
Queue weights and queue depths are configurable.

Drop thresholds can be configured as tail drop or WRED.

Queuing on Catalyst Switches (Cont.)

Catalyst 2950 Switches

- 4 transmit queues (1P3Q or 4Q)
- Need to configure PQ and ensure that CoS 5 traffic is assigned to the PQ
 - Configurable PQ for **queue 4**
 - Configurable CoS to specific queue
 - Configurable queue weight



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.5

QoS is enabled by default on Cisco Catalyst 2950 Series switches, and the default scheduling method is strict priority. Strict priority scheduling is based on the priority of queues. Packets in the high-priority queue always transmit first; packets in the low-priority queue do not transmit until all the high-priority queues become empty.

CoS values can be assigned to queues during configuration. The default CoS-to-queue assignment is:

- CoS 6 to 7 placed in queue 4
- CoS 4 to 5 placed in queue 3
- CoS 2 to 3 placed in queue 2
- CoS 0 to 1 placed in queue 1

Catalyst 2950 switches support PQ, WRR scheduling, and WRR with a priority queue, as follows:

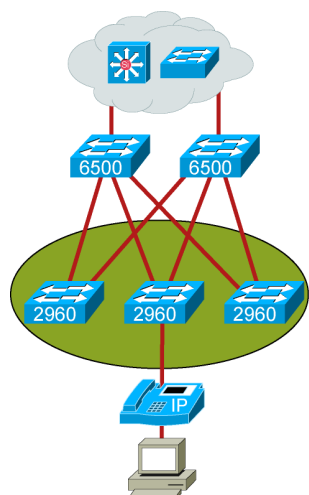
- The WRR scheduling algorithm ensures that lower-priority packets are not entirely starved for bandwidth and are serviced without compromising the priority settings administered by the network manager.
- WRR with a priority queue ensures that higher-priority packets will always get serviced first, ahead of other traffic in lower priority queues. The priority queue is defined as queue 4.

Queue weights are configurable.

Queuing on Catalyst Switches (Cont.)

Catalyst 2960 Switches

- 2 ingress queues (1P1Q3T or 2Q3T).
- 4 egress queues per port (1P3Q3T or 4Q3T).
- Enhanced version of tail-drop called "weighted tail drop" used on ingress and egress queues.
- Ingress and egress queues serviced by Shaped Round Robin.



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.6

Cisco Catalyst 2960 switches have queues at specific points to help prevent congestion. Because the total inbound bandwidth of all ports can exceed the bandwidth of the internal ring, packets are sent to ingress queues after being classified, policed, and marked and before being forwarded into the switch fabric. Because multiple ingress ports can simultaneously send packets to an egress port and cause congestion, packets are forwarded from the internal ring to outbound queues. Catalyst 2960 switches have the following queues:

- **Two configurable ingress queues for user traffic:** One of the ingress queues can be the priority queue.
 - WTD is used as the congestion-avoidance mechanism for managing the queue lengths and providing drop precedences for different traffic classifications. To use WTD, the switch must be running the LAN Base image.
 - Shaped round robin (SRR) is used as the scheduling service for specifying the rate at which packets are sent to the internal ring. SRR services the priority queue for its configured share before servicing the other queue. Sharing is the only supported mode on ingress queues. To use ingress queueing, the switch must be running the LAN Base image.
- **Four egress queues per port:** One of the egress queues (queue 1) can be the expedite queue. The expedite queue is a priority queue, and it is serviced until empty before the other queues are serviced.
 - WTD is used as the congestion-avoidance mechanism.
 - SRR is used as the scheduling service for specifying the rate at which packets are dequeued to the egress interface. Shaping or sharing is supported on egress queues. Shaped egress queues are guaranteed but limited to using a share of port bandwidth. Shared egress queues are also guaranteed a configured share of bandwidth, but can use more than the guarantee if other queues become empty and do not use their share of the bandwidth. To use egress queueing, the switch must be running the LAN Base image.

On the Cisco Catalyst 2960 Series switches, you can prioritize traffic by placing packets with particular DSCP or CoS values into certain queues, by allocating a large queue size, or by servicing the queue more frequently. You can also adjust queue thresholds so that packets with lower priorities are dropped.

By default, QoS is disabled on Cisco Catalyst 2960 Series switches. There is no concept of trusted or untrusted ports because the packets are not modified; the CoS, DSCP, and IP precedence values in the packet are not changed. Traffic is switched in pass-through mode. (Packets are switched without any rewrites and classified as best effort without any policing.)

When you enable QoS with the **mls qos** global configuration command and all other QoS settings are set to their defaults, traffic is classified as best effort without any policing. No policy maps are configured. The default port trust state on all ports is untrusted, and the default ingress and egress queue settings are in effect.

Note	QoS is also disabled by default on Catalyst 3560 and 3750 switches, and the mls qos command is used to globally enable it. As with the Catalyst 2960 switches, WTD and SRR are used for congestion-avoidance and scheduling on these switches.

The default ingress and egress queue settings are described in the tables below.

Default Ingress Queue Configuration

Feature	Queue 1	Queue 2
Buffer allocation	90 percent	10 percent
Bandwidth allocation ¹	4	4
Priority queue bandwidth ²	0	10
WTD drop threshold 1	100 percent	100 percent
WTD drop threshold 2	100 percent	100 percent

¹The bandwidth is equally shared between the queues. SRR sends packets in shared mode only.

²Queue 2 is the priority queue. SRR services the priority queue for its configured share before servicing the other queue.

Default Egress Queue Configuration

Feature	Queue 1	Queue 2	Queue 3	Queue 4
Buffer allocation	25 percent	25 percent	25 percent	25 percent
WTD drop threshold 1	100 percent	200 percent	100 percent	100 percent
WTD drop threshold 2	100 percent	200 percent	100 percent	100 percent
Reserved threshold	50 percent	50 percent	50 percent	50 percent
Maximum threshold	400 percent	400 percent	400 percent	400 percent
SRR shaped weights (absolute) ¹	25	0	0	0
SRR shared weights ²	25	25	25	25

¹A shaped weight of zero means that this queue is operating in shared mode.

²One-quarter of the bandwidth is allocated to each queue.

Weighted Round Robin

This topic describes how WRR works on a Catalyst 2950 switch.

Weighted Round Robin

- WRR overcomes the problem of having PQ starving out the lower-priority queues.
- WRR scheduling prevents queues with a lower weight from being completely starved during periods of heavy high-priority traffic.
- Different weights are assigned to each queue.
- For example, in one scheduling round, the WRR scheduler will transmit:
 - Three frames from a queue assigned weight 3.
 - Four frames from a queue assigned weight 4.
- WRR with an expedite queue: When WRR is configured on a Catalyst 2950, the option exists to configure queue 4 as a priority queue—an “expedite queue.”

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.7

WRR scheduling requires that you specify a number that indicates the importance (weight) of the queue relative to the other CoS queues. WRR scheduling prevents the low-priority queues from being completely neglected during periods of high-priority traffic. The WRR scheduler transmits some packets from each queue in turn. The number of packets it sends corresponds to the relative importance of the queue. For example, if one queue has a weight of 3 and another has a weight of 4, three packets are sent from the first queue for every four that are sent from the second queue. By using this scheduling, low-priority queues have the opportunity to send packets even though the high-priority queues are not empty.

WRR scheduling with an expedite priority queue (also referred to as strict PQ) uses one of the egress queues as an expedite queue (queue 4 on a Catalyst 2950 switch). The remaining queues participate in WRR. When the expedite queue is configured, it is a priority queue and is serviced until it is empty before the other queues are serviced by WRR scheduling. Actions at the egress interface include queuing and scheduling, as follows:

- Queuing evaluates the internal DSCP and determines in which of the four egress queues to place the packet. The DSCP value is mapped to a CoS value, which selects one of the queues.
- Scheduling services the four egress queues based on their configured WRR weights and thresholds. One of the queues can be the expedite queue, which is serviced until empty before the other queues are serviced.

With WRR, lower-priority queues have the opportunity to transmit packets even though the high-priority queues have not been emptied. With WRR with an expedite queue, one queue (queue 4 on the Catalyst 2950 and 3550 switches) can be configured as an expedite priority queue. All traffic from the expedite queue must be serviced before the remaining three queues are serviced. The expedite queue can be used to ensure that voice traffic incurs minimal delay and no drops.

Configuring CoS-to-Queue Mappings for PQ on Catalyst 2950 Switches

This topic describes the commands required to configure CoS-to-queue mappings for PQ on the Cisco Catalyst 2950 switch.

Configuring PQ on Catalyst 2950 Switches

```
switch(config)#
```

```
wrr-queue cos-map quid cos1...cosn
```

- Assigns CoS values to CoS priority queues.
- **quid**: Specifies the queue ID of the CoS priority queue. (Ranges are 1 to 4, where 1 is the lowest CoS priority queue.)
- **cos1...cosn**: Specifies the CoS values that are mapped to the queue ID.
- Default ID values are:

Queue ID	CoS Values
1	0, 1
2	2, 3
3	4, 5
4	6, 7

© 2009 Cisco Systems, Inc. All rights reserved.CoS v2.3-5.8

To configure CoS-to-queue mappings for PQ on the Catalyst 2950 switch, specify the queue ID of the CoS priority queue. Ranges are 1 to 4 where 1 is the lowest CoS priority queue. Then, specify the CoS values that are mapped to the queue ID.

The syntax for the **wrr-queue cos-map** command is as follows:

```
wrr-queue cos-map quid cos1...cosn
```

Syntax Description

Parameter	Description
<i>quid</i>	The queue ID of the CoS priority queue. Ranges are 1 to 4 where 1 is the lowest CoS priority queue.
<i>cos1...cosn</i>	The CoS values that are mapped to the queue ID.

The default CoS-to-priority queue assignments are shown in this table.

Queue	1	2	3	4
CoS Values	0, 1	2, 3	4, 5	6, 7

Configuring WRR on Catalyst 2950 Switches

This topic describes the commands required to configure WRR on the Catalyst 2950 switch.

Configuring WRR on Catalyst 2950 Switches

```
switch(config)#  
wrr-queue bandwidth weight1...weight4
```

- Assigns WRR weights to the four egress queues.
- Ranges for the WRR values:
 - For **weight1**, **weight2**, and **weight3**, the range is 1 to 255.
 - For **weight4**, the range is 0 to 255 (when **weight4** is set to 0, queue 4 is configured as the expedite queue).

```
wrr-queue bandwidth 20 1 80 0  
no wrr-queue cos-map  
wrr-queue cos-map 1 0 1 2 4  
wrr-queue cos-map 3 3 6 7  
wrr-queue cos-map 4 5
```

© 2003 Cisco Systems, Inc. All rights reserved. QoS v2.3-5.9

Use the **wrr-queue bandwidth** global configuration command to assign WRR weights to the four CoS priority queues on the Catalyst 2950 switch. Use the **no** form of this command to disable the WRR scheduler and enable the strict priority scheduler.

For weight 1, weight 2, and weight 3, the range is 1 to 255. The range for weight 4 is 0 to 255. Queues 1, 2, and 3 can be configured for WRR scheduling and queue 4 can be configured for strict priority scheduling. To configure queue 4 as the expedite queue, set weight 4 to 0. When queue 4 is empty, packets from queues 1, 2, and 3 are sent according to the assigned WRR weights.

The syntax for the **wrr-queue bandwidth** command is as follows:

```
wrr-queue bandwidth weight1...weight4
```

Syntax Description

Parameter	Description
<i>weight1...weight4</i>	The ratio of weight 1, weight 2, weight 3, and weight 4 determines the weights of the WRR scheduler.

Note In Cisco IOS Software releases earlier than Release 12.1(12c)EA1, the range for all queues is 1 to 255.

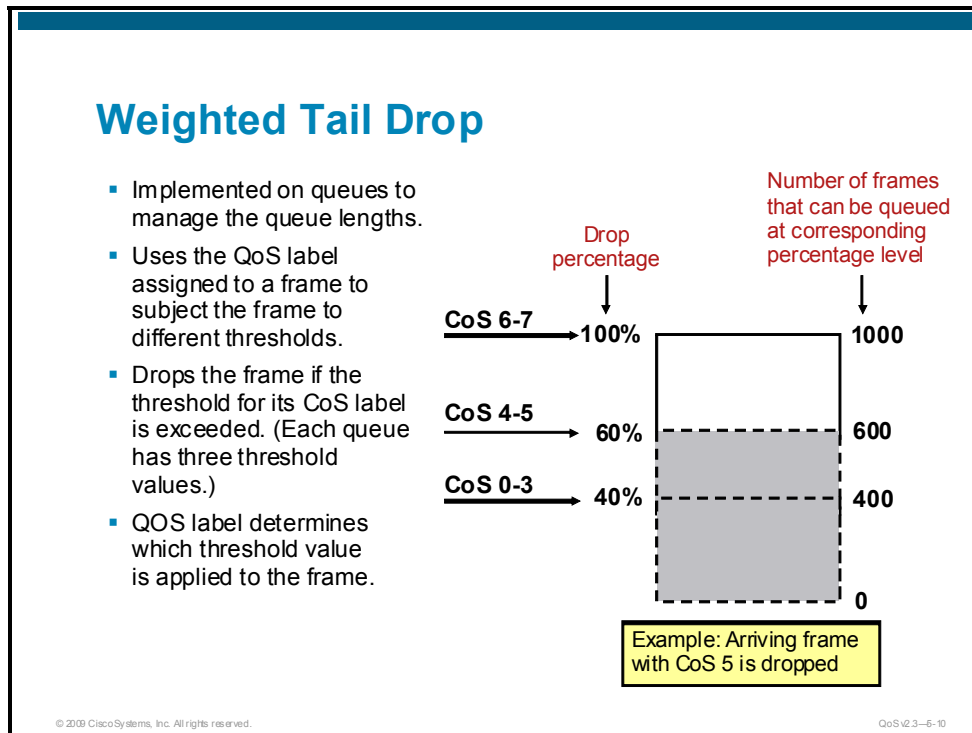
In the example for the Catalyst 2950 switch, this configuration has been made:

- The interface is set to Gigabit Ethernet 0/12.
- Queue bandwidth (queue weight) is set to these weights:
 - Queue 1: 20
 - Queue 2: 1
 - Queue 3: 80
 - Queue 4: 0 (because this is 0, this is the expedite queue)
- The CoS map is set to its default settings.
- CoS is mapped to queues according to the following:
 - Queue 1: CoS 0, 1, 2, 4
 - Queue 2: No CoS assigned
 - Queue 3: CoS 3, 6, 7
 - Queue 4: CoS 5 (voice traffic goes to the expedite queue)

Note	This is the AutoQoS configuration for the Catalyst 2950.

Weighted Tail Drop

This topic describes how WTD works on a Catalyst 2960 switch.



On Cisco Catalyst 2960 Series switches, both the ingress and egress queues use an enhanced version of the tail-drop congestion-avoidance mechanism called weighted tail drop (WTD). WTD is implemented on queues to manage the queue lengths and to provide drop precedences for various traffic classifications.

As a frame is enqueued to a particular queue, WTD uses the QoS label that is assigned to the frame to subject the frame to different thresholds. If the threshold is exceeded for that QoS label (the space available in the destination queue is less than the size of the frame), the switch drops the frame.

Each queue has three drop thresholds: two configurable (explicit) WTD thresholds and one nonconfigurable (implicit) threshold preset to the queue-full state. The QoS label determines which of the three threshold values is applied to the frame.

You can assign the two configurable WTD threshold percentages for threshold ID 1 and ID 2 to the ingress queues by using the **mls qos srr-queue input threshold** command. Each threshold value is a percentage of the total number of allocated buffers for the queue. The drop threshold for threshold ID 3 is preset to the queue-full state, and you cannot modify it.

The figure shows an example of WTD operating on a queue whose size is 1000 frames. Three drop percentages are configured: 40 percent (400 frames), 60 percent (600 frames), and 100 percent (1000 frames). These percentages mean that up to 400 frames can be queued at the 40-percent threshold, up to 600 frames at the 60-percent threshold, and up to 1000 frames at the 100-percent threshold.

In this example, CoS values 6 and 7 have a greater importance than the other CoS values, and they are assigned to the 100-percent drop threshold (queue-full state). CoS values 4 and 5 are assigned to the 60-percent threshold, and CoS values 0 to 3 are assigned to the 40-percent threshold.

Suppose the queue is already filled with 600 frames, and a new frame arrives. It contains CoS values 4 and 5 and is subjected to the 60-percent threshold. If this frame is added to the queue, the threshold will be exceeded, so the switch drops it.

Configuring WTD on Catalyst 2960 Switches

This topic describes the commands required to configure WTD on the Catalyst 2960 switch.

Allocating Buffer Space to and Setting WTD Thresholds for an Egress Queue Set

```
switch(config)#
```

```
mls qos queue-set output qset-id threshold queue-id drop-  
threshold1 drop-threshold2 reserved-threshold maximum-  
threshold
```

- Configures the weighted tail-drop (WTD) thresholds, guarantees the availability of buffers, and configures the maximum memory allocation to a queue-set (four egress queues per port).

```
switch(config)#
```

```
mls qos queue-set output qset-id buffers allocation1 ...  
allocation4
```

- Allocates buffers to a queue-set (four egress queues per port).

```
switch(config-if)#
```

```
queue-set qset-id
```

- Maps a port to a queue set.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6-11

You can guarantee the availability of buffers, set WTD thresholds, and configure the maximum allocation for a queue set by using the **mls qos queue-set output threshold** command. A queue set defines all the characteristics of the four egress queues per port.

Each threshold value is a percentage of the queues allocated buffers, which you specify by using the **mls qos queue-set output buffers** command. The queues use WTD to support distinct drop percentages for different traffic classes.

Note The egress queue default settings are suitable for most situations. You should change them only when you have a thorough understanding of the egress queues and if the default settings do not meet your QoS solution.

Complete the following steps to configure the memory allocation and to drop thresholds for a queue set:

- Step 1** Allocate buffers to a queue set. By default, all allocation values are equally mapped among the four queues (25, 25, 25, 25). Each queue has one-fourth of the buffer space. Allocate buffers according to the importance of the traffic. For example, give a large percentage of the buffer to the queue with the highest-priority traffic.

- Step 2** Configure the WTD thresholds, guarantee the availability of buffers, and configure the maximum memory allocation for the queue set (four egress queues per port). By default, the WTD thresholds for queues 1, 3, and 4 are set to 100 percent. The thresholds for queue 2 are set to 200 percent. The reserved thresholds for queues 1, 2, 3, and 4 are set to 50 percent. The maximum thresholds for all queues are set to 400 percent.
- Step 3** Map a port to a queue set.

The syntax for the **mls qos queue-set output buffers** command is as follows:

```
mls qos queue-set output qset-id buffers allocation1 ...  
allocation4
```

Syntax Description

Parameter	Description
<i>qset-id</i>	ID of the queue set. Each port belongs to a queue set, which defines all the characteristics of the four egress queues per port. The range is 1 to 2.
<i>allocation1...allocation4</i>	Buffer space allocation (percentage) for each queue (four values for queues 1 to 4). For <i>allocation1</i> , <i>allocation3</i> , and <i>allocation4</i> , the range is 0 to 99. For <i>allocation2</i> , the range is 1 to 100 (including the CPU buffer). Separate each value with a space.

The syntax for the **mls qos queue-set output threshold** command is as follows:

```
mls qos queue-set output qset-id threshold queue-id drop-  
threshold1 drop-threshold2 reserved-threshold maximum-  
threshold
```

Syntax Description

Parameter	Description
<i>qset-id</i>	ID of the queue set. The range is 1 to 2.
<i>queue-id</i>	Specific queue in the queue set on which the command is performed. The range is 1 to 4.
<i>drop-threshold1 drop-threshold2</i>	Two WTD thresholds expressed as a percentage of the allocated memory of the queue. The range is 1 to 3200 percent.
<i>reserved-threshold</i>	Amount of memory to be guaranteed (reserved) for the queue and expressed as a percentage of the allocated memory. The range is 1 to 100 percent.
<i>maximum-threshold</i>	Enable a queue in the full condition to get more buffers than are reserved for it. This is the maximum memory the queue can have before the packets are dropped. The range is 1 to 3200 percent.

The syntax for the **queue-set** command is as follows:

```
queue-set qset-id
```

Syntax Description

Parameter	Description
<i>qset-id</i>	ID of the queue set. Each port belongs to a queue set, which defines all the characteristics of the four egress queues per port. The range is 1 to 2.

Allocating Buffer Space to and Setting WTD Thresholds for an Egress Queue Set (Cont.)

```
mls qos queue-set output 2 buffers 40 20 20 20
mls qos queue-set output 2 threshold 2 40 60 100 200
interface gigabitethernet0/1
queue-set 2
```

- Maps a port to queue set 2
 - Allocates 40% of the buffer space to egress queue 1 and 20% to egress queues 2, 3, and 4.
 - Configures the drop thresholds for queue 2 to 40% (threshold 1) and 60% (threshold 2) of the allocated memory.
 - Guarantees 100% of the allocated memory to queue 2.
 - Configures 200% as the maximum memory that queue 2 can have before packets are dropped.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-12

The example in the figure shows how to map a port to queue set 2. It allocates 40 percent of the buffer space to egress queue 1 and 20 percent to egress queues 2, 3, and 4. It configures the drop thresholds for queue 2 to 40 and 60 percent of the allocated memory, guarantees (reserves) 100 percent of the allocated memory, and configures 200 percent as the maximum memory that this queue can have before packets are dropped.

Mapping DSCP or CoS Values to an Egress Queue and to a Threshold ID

switch(config)#

```
mls qos srr-queue output dscp-map queue queue-id  
{dscp1...dscp8 | threshold threshold-id dscp1...dscp8}
```

- Maps DSCP values to an egress queue or maps DSCP values to a queue and to a threshold ID.

switch(config)#

```
mls qos srr-queue output cos-map queue queue-id {cos1...cos8  
| threshold threshold-id cos1...cos8}
```

- Maps CoS values to an egress queue or maps CoS values to a queue and to a threshold ID.

```
mls qos srr-queue output cos-map queue 1 threshold 1 0 1 2 3
```

- Maps CoS values 0 to 3 to egress queue 1 and to threshold 1.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5-13

You can prioritize traffic by placing packets with particular DSCPs or CoS values into certain queues and adjusting the queue thresholds so that packets with lower priorities are dropped.

Note

The egress queue default settings are suitable for most situations. You should change them only when you have a thorough understanding of the egress queues and if the default settings do not meet your QoS solution.

Use the **mls qos srr-queue output dscp-map** command or the **mls qos srr-queue output cos-map** command to map DSCP or CoS values to an egress queue and to a threshold ID. By default, DSCP values 0-15 are mapped to queue 2 and threshold 1. DSCP values 16-31 are mapped to queue 3 and threshold 1. DSCP values 32-39 and 48-63 are mapped to queue 4 and threshold 1. DSCP values 40-47 are mapped to queue 1 and threshold 1.

By default, CoS values 0 and 1 are mapped to queue 2 and threshold 1. CoS values 2 and 3 are mapped to queue 3 and threshold 1. CoS values 4, 6, and 7 are mapped to queue 4 and threshold 1. CoS value 5 is mapped to queue 1 and threshold 1.

The syntax for the **mls qos srr-queue output dscp-map** command is as follows:

```
mls qos srr-queue output dscp-map queue queue-id  
{dscp1...dscp8 | threshold threshold-id dscp1...dscp8}
```

Syntax Description

Parameter	Description
queue <i>queue-id</i>	Specifies a queue number. For <i>queue-id</i> , the range is 1 to 4.
<i>dscp1...dscp8</i>	Maps DSCP values to an egress queue. For <i>dscp1...dscp8</i> , enter up to eight values, and separate each value with a space. The range is 0 to 63.
threshold <i>threshold-id</i> <i>dscp1...dscp8</i>	Maps DSCP values to a queue threshold ID. For <i>threshold-id</i> , the range is 1 to 3. For <i>dscp1...dscp8</i> , enter up to eight values, and separate each value with a space. The range is 0 to 63.

The syntax for the **mls qos srr-queue output cos-map** command is as follows:

```
mls qos srr-queue output cos-map queue queue-id {cos1...cos8 |  
threshold threshold-id cos1...cos8}
```

Syntax Description

Parameter	Description
queue <i>queue-id</i>	Specifies a queue number. For <i>queue-id</i> , the range is 1 to 4.
<i>cos1...cos8</i>	Maps CoS values to an egress queue. For <i>cos1...cos8</i> , enter up to eight values, and separate each value with a space. The range is 0 to 7.
threshold <i>threshold-id</i> <i>cos1...cos8</i>	Maps CoS values to a queue threshold ID. For <i>threshold-id</i> , the range is 1 to 3. For <i>cos1...cos8</i> , enter up to eight values, and separate each value with a space. The range is 0 to 7.

Shaped Round Robin

This topic describes how SRR works on a Catalyst 2960 switch.

Shaped Round Robin

- Controls the rate at which packets are sent.
 - On ingress queues, sends packets to internal ring.
 - On egress queues, sends packets to egress port.
- Has two modes:
 - Shaped mode (for egress queues only)
 - Guarantees queues a percentage of bandwidth and rate-limits them to that amount.
 - Provides a more even flow of traffic over time.
 - Absolute value of a weight is used to compute bandwidth available for the queues.
 - Shared mode
 - Queues share bandwidth according to configured weights.
 - Bandwidth is guaranteed; queues can also use unused bandwidth allocated to another queue.
 - Ratio of weights controls frequency of dequeuing; absolute values are meaningless.
- Shaping and sharing are configured on a per-interface basis.

© 2009 Cisco Systems, Inc. All rights reserved.

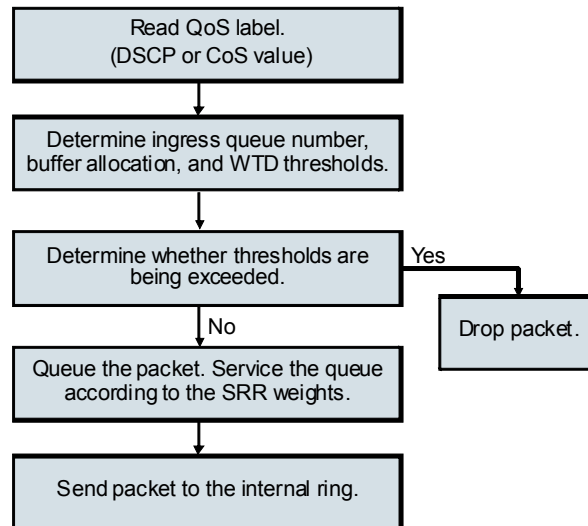
QoS v2.3--5-14

On Cisco Catalyst 2960 Series switches, both the ingress and egress queues are serviced by SRR, which controls the rate at which packets are sent. On the ingress queues, SRR sends packets to the internal ring. On the egress queues, SRR sends packets to the egress port.

In shaped mode, the egress queues are guaranteed a percentage of the bandwidth, and they are rate-limited to that amount. Shaped traffic does not use more than the allocated bandwidth even if the link is idle. Shaping provides a more even flow of traffic over time and reduces the peaks and valleys of bursty traffic. With shaping, the absolute value of each weight is used to compute the bandwidth available for the queues.

In shared mode, the queues share the bandwidth among them according to the configured weights. The bandwidth is guaranteed at this level but not limited to it. For example, if a queue is empty and no longer requires a share of the link, the remaining queues can expand into the unused bandwidth and share it among them. With sharing, the ratio of the weights controls the frequency of dequeuing; the absolute values are meaningless. Shaping and sharing are configured per interface. Each interface can be uniquely configured.

Queuing and Scheduling on Ingress Queues



© 2009 Cisco Systems, Inc. All rights reserved.

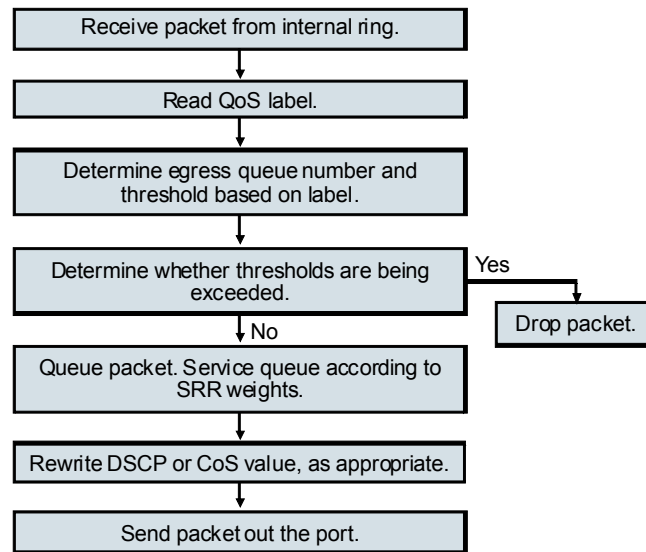
QoS v2.3-6-15

The figure shows the queuing and scheduling flowchart for ingress ports.

You define the ratio (allocate the amount of space) with which to divide the ingress buffers between the two ingress queues by using the **mls qos srr-queue input buffers** command. The buffer allocation together with the bandwidth allocation control how much data can be buffered and sent before packets are dropped. You allocate bandwidth as a percentage by using the **mls qos srr-queue input bandwidth** command. The ratio of the weights is the ratio of the frequency in which the SRR scheduler sends packets from each queue.

SRR services the ingress priority queue for its configured weight as specified by the **bandwidth** keyword in the **mls qos srr-queue input priority-queue** command. SRR then shares the remaining bandwidth with both ingress queues and services them as specified by the weights configured with the **mls qos srr-queue input bandwidth** command.

Queuing and Scheduling on Egress Queues



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--5-16

The four egress queues are configured by a queue set. All traffic leaving an egress port flows through one of these four queues and is subjected to a threshold based on the QoS label assigned to the packet.

Queuing and Scheduling on Egress Queues (Cont.)

- Buffer space divided between common pool and reserved pool.
- Switch reserves a minimum amount of buffers for each egress queue.
- Switch detects
 - Whether target queue is under limit or over limit.
 - Whether common pool is empty (no free buffers) or not empty (free buffers).
- If queue is not over limit, the switch can allocate buffer space
 - From the reserved pool.
 - From the common pool if it is not empty.
- If no free buffers in common pool or if queue over limit, switch drops frame.



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-17

The figure shows the egress queue buffer. The buffer space is divided between the common pool and the reserved pool. The switch uses a buffer allocation scheme to reserve a minimum amount of buffers for each egress queue, to prevent any queue or port from consuming all the buffers and depriving other queues, and to control whether to grant buffer space to a requesting queue. The switch detects whether the target queue has not consumed more buffers than its reserved amount (under limit), whether it has consumed all of its maximum buffers (over limit), and whether the common pool is empty (no free buffers) or not empty (free buffers). If the queue is not over limit, the switch can allocate buffer space from the reserved pool or from the common pool if it is not empty. If there are no free buffers in the common pool or if the queue is over limit, the switch drops the frame.

You guarantee the availability of buffers, set drop thresholds, and configure the maximum memory allocation for a queue-set by using the **mls qos queue-set output threshold** command. Each threshold value is a percentage of the memory allocated to the queue. You can allocate memory to a queue by using the **mls qos queue-set output buffers** command. The sum of all the allocated buffers represents the reserved pool, and the remaining buffers are part of the common pool. Through buffer allocation, you can ensure that high-priority traffic is buffered. For example, if the buffer space is 400, you can allocate 70 percent of it to queue 1 and 10 percent to queues 2 through 4. Queue 1 then has 280 buffers allocated to it, and queues 2 through 4 each have 40 buffers allocated to them.

You can guarantee that the allocated buffers are reserved for a specific queue in a queue set. For example, if there are 100 buffers for a queue, you can reserve 50 percent (50 buffers). The switch returns the remaining 50 buffers to the common pool. You also can enable a queue in the full condition to obtain more buffers than are reserved for it by setting a maximum threshold. The switch can allocate the needed buffers from the common pool if the common pool is not empty.

SRR services each queue set in shared or shaped mode. You assign shared or shaped weights to the port by using the **srr-queue bandwidth share** command or the **srr-queue bandwidth shape** command. The buffer allocation together with the SRR weight ratios control how much data can be buffered and sent before packets are dropped. The weight ratio is the ratio of the frequency with which the SRR scheduler sends packets from each queue.

All four queues participate in the SRR unless the expedite queue is enabled, in which case the first bandwidth weight is ignored and is not used in the ratio calculation. Use the **priority-queue out** command to enable the expedite queue.

Configuring SRR on Catalyst 2960 Switches

This topic describes the commands required to configure SRR on the Catalyst 2960 switch.

Allocating Bandwidth to Ingress Queues and Configuring the Priority Queue

```
switch(config)#
```

```
mls qos srr-queue input priority-queue queue-id  
bandwidth weight
```

- Configures the ingress priority queue and guarantees bandwidth on the internal ring if the ring is congested.

```
switch(config)#
```

```
mls qos srr-queue input bandwidth weight1 weight2
```

- Assigns SRR weights to an ingress queue.

```
mls qos srr-queue input priority-queue 2 bandwidth 0  
mls qos srr-queue input bandwidth 25 75
```

- Disables priority queuing.
- Assigns bandwidth to the ingress queues; configures the switch to service queue 2 three times as often as queue 1.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6-18

Use the **mls qos srr-queue input priority-queue** command to configure the ingress priority queue and to guarantee bandwidth on the internal ring if the ring is congested. SRR services the priority queue for its configured weight as specified by the **bandwidth** keyword in this command. You should use the priority queue only for traffic that needs to be expedited, such as voice traffic, which needs minimum delay and jitter. The priority queue is guaranteed part of the bandwidth to reduce the delay and jitter under heavy network traffic on an oversubscribed ring (when there is more traffic than the backplane can carry, and the queues are full and dropping frames).

After servicing the ingress priority queue for its configured weight, SRR shares the remaining bandwidth between the two ingress queues, servicing them according to the weights you specify with the **mls qos srr-queue input bandwidth** command. The ratio of the weights is the ratio of the frequency with which the SRR scheduler sends packets from each queue. The bandwidth and the buffer allocation control how much data can be buffered before packets are dropped.

The syntax for the **mls qos srr-queue input priority-queue** command is as follows:

```
mls qos srr-queue input priority-queue queue-id bandwidth  
weight
```

Syntax Description

Parameter	Description
<i>queue-id</i>	Ingress queue ID. The range is 1 to 2. By default, the priority queue is queue 2.
bandwidth <i>weight</i>	Bandwidth percentage of the internal ring. The range is 0 to 40. By default, 10 percent of the bandwidth is allocated to the priority queue.

The syntax for the **mls qos srr-queue input bandwidth** command is as follows:

```
mls qos srr-queue input bandwidth weight1 weight2
```

Syntax Description

Parameter	Description
<i>weight1 weight2</i>	Ratio of weight1 and weight2 determines the ratio of the frequency with which the SRR scheduler dequeues packets from ingress queues 1 and 2. The range is 1 to 100. Separate each value with a space. The default setting for weight1 and weight2 is 4; one-half of the bandwidth is equally shared between the two queues.

Configuring Weights on Egress Queues and Enabling the Expedite Queue

```
switch(config-if)#
```

```
srr-queue bandwidth share weight1 weight2 weight3 weight4
```

- Assigns the shared weights and enables bandwidth sharing on the four egress queues mapped to a port.

```
switch(config-if)#
```

```
srr-queue bandwidth shape weight1 weight2 weight3 weight4
```

- Assigns the shaped weights and enables bandwidth shaping on the four egress queues mapped to a port.

```
switch(config-if)#
```

```
priority-queue out
```

- Enables the egress expedite queue on a port.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-10

To assign shared weights and to enable bandwidth sharing on the four egress queues mapped to a port, use the **srr-queue bandwidth share** command in interface configuration mode. By default, all four weights are 25; one-fourth of the bandwidth is allocated to each queue.

Use the **srr-queue bandwidth shape** command to assign SRR shaped weights to the egress queues. By default, weight1 is set to 25; weight2, weight3, and weight4 are set to 0, and these queues are in shared mode.

Use the **priority-queue** command to enable the egress expedite queue, which is disabled by default. When you configure this command, the SRR weight and queue size ratios are affected because there is one less queue participating in SRR. This means that *weight1* in the **srr-queue bandwidth shape** command or the **srr-queue bandwidth share** command is ignored; it is not used in the ratio calculation.

The shaped mode overrides the shared mode. When configuring queues for the same port for both shaping and sharing, make sure that you configure the lowest-numbered queue for shaping. If you set a shaped queue weight to 0, the queue participates in shared mode and the weights specified with the **srr-queue bandwidth share** command come into effect for the queue.

The syntax for the **srr-queue bandwidth share** command is as follows:

```
srr-queue bandwidth share weight1 weight2 weight3 weight4
```

Syntax Description

Parameter	Description
<i>weight1 weight2 weight3 weight4</i>	The ratios of weight1, weight2, weight3, and weight4 specify the ratio of the frequency with which the SRR scheduler dequeues packets. Separate each value with a space. The range is 1 to 255.

The syntax for the **srr-queue bandwidth shape** command is as follows:

```
srr-queue bandwidth shape weight1 weight2 weight3 weight4
```

Syntax Description

Parameter	Description
<i>weight1 weight2 weight3 weight4</i>	Specify the weights to specify the percentage of the port that is shaped. The inverse ratio (1/weight) specifies the shaping bandwidth for this queue. Separate each value with a space. The range is 0 to 65535.

Example: Configuring Weights on Egress Queues and Enabling the Expedite Queue

- Configures the queues for the same port for both shaping and sharing.
- The bandwidth weight for queue 1 is 1/25 or 4%.
- Because the weight ratios for queues 2, 3, and 4 are set to 0, they operate in shared mode.
- The bandwidth ratio allocated for the queues in shared mode is
 - Queue 2: $20/(20+25+25)$ or 28.6%.
 - Queue 3: $25/(20+25+25)$ or 35.7%.
 - Queue 4: $25/(20+25+25)$ or 35.7%.
- Disables the egress expedite queue after the SRR shaped and shared weights are configured.

```
interface gigabitethernet0/1
  srr-queue bandwidth shape 25 0 0 0
  srr-queue bandwidth share 30 20 25 25
  no priority-queue out
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-20

The figure shows an example in which queues for the same port are configured for both shaping and sharing. Because the shaped mode overrides the shared mode, the bandwidth weight for queue 1 is 1/25, or 4%. The weight ratios for queues 2, 3, and 4 are set to 0, so those queues operate in shared mode. After the SRR shaped and shared weights are configured, the egress expedite queue is disabled.

Limiting the Bandwidth on an Egress Interface

```
switch(config-if) #
```

```
srr-queue bandwidth limit weight1
```

- Limits the maximum output on a port.

```
interface gigabitethernet0/1
```

```
srr-queue bandwidth limit 80
```

- Limits the bandwidth on a port to 80%.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3—5-21

You can limit the bandwidth on an egress port. For example, if a customer pays for only a small percentage of a high-speed link, you can limit the bandwidth to that amount.

Use the **srr-queue bandwidth limit** command to specify the percentage of the port speed to which the port should be limited. The range is 10 to 90. By default, the port is not rate-limited and is set to 100 percent.

In the figure, the bandwidth on the gigabitethernet0/1 interface is limited to 80 percent. This means that the port is idle 20 percent of the time. The line rate drops to 80 percent of the connected speed, which is 800 Mb/s. These values are not exact, because the hardware adjusts the line rate in increments of six.

The syntax for the **srr-queue bandwidth limit** command is as follows:

```
srr-queue bandwidth limit weight1
```

Syntax Description

Parameter	Description
<i>weight1</i>	Percentage of the port speed to which the port should be limited. The range is 10 to 90.

Monitoring Queuing on Catalyst 2950 and 2960 Switches

This topic describes the commands required to monitor queuing on Catalyst 2950 and 2960 switches.

Monitoring Queuing on Catalyst 2950 and 2960 Switches

```
switch>
show mls qos maps [cos-dscp | dscp-cos]
```

- Displays QoS mapping information.
- This command is available with enhanced software image switches.

```
switch> show mls qos maps

Dscp-cos map:
dscp: 0 8 10 16 18 24 26 32 34 40 46 48 56
-----
cos:  0 1  1  2  2  3  3  4  4  5  5  6  7

Cos-dscp map:
cos:  0 1 2 3 4 5 6 7
-----
dscp: 0 8 16 24 32 40 48 56
```

(show mls qos maps command output from a Catalyst 2950 switch)

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-6-22

Use the **show mls qos maps** command to display QoS mapping information on the Catalyst 2950 and 2960 switches. Maps are used to generate an internal DSCP value, which represents the priority of the traffic. The figure shows output of this command on a Catalyst 2950 switch.

The **show mls qos maps** command is available only if the Catalyst 2950 switch is running the Enhanced Image (EI) software.

If the **show mls qos maps** command is used without any keywords, it will display all maps. The syntax for the **show mls qos maps** command is as follows:

```
show mls qos maps [cos-dscp | dscp-cos]
```

Syntax Description

Parameter	Description
cos-dscp	(Optional) Displays CoS-to-DSCP map.
dscp-cos	(Optional) Displays DSCP-to-CoS map.

The syntax for the **show mls qos maps** command on Catalyst 2960 Series switches running IOS Software Release 12.2(46)SE is as follows:

```
show mls qos maps [cos-dscp | cos-input-q | cos-output-q |  
dscp-cos | dscp-input-q | dscp-mutation dscp-mutation-name |  
dscp-output-q | ip-prec-dscp | policed-dscp] [ | {begin |  
exclude | include} expression]
```

Syntax Description

Parameter	Description
cos-dscp	(Optional) Displays CoS-to-DSCP map.
cos-input-q	(Optional) Display the CoS input queue threshold map.
cos-output-q	(Optional) Display the CoS output queue threshold map.
dscp-cos	(Optional) Displays DSCP-to-CoS map.
dscp-input-q	(Optional) Display the DSCP input queue threshold map.
dscp-mutation <i>dscp-mutation-name</i>	(Optional) Display the specified DSCP-to-DSCP-mutation map.
dscp-output-q	(Optional) Display the DSCP output queue threshold map.
ip-prec-dscp	(Optional) Display the IP-precedence-to-DSCP map.
policed-dscp	(Optional) Display the policed-DSCP map.
begin	(Optional) Display begins with the line that matches the <i>expression</i> .
exclude	(Optional) Display excludes lines that match the <i>expression</i> .
include	(Optional) Display includes lines that match the specified <i>expression</i> .
<i>expression</i>	Expression in the output to use as a reference point.

Monitoring Queuing on Catalyst 2950 Switches

```
switch>
```

```
show wrr-queue bandwidth
```

- Displays the WRR bandwidth allocation for the CoS priority queues.

```
switch> show wrr-queue bandwidth
WRR Queue : 1 2 3 4
Bandwidth : 10 20 30 40
```

```
switch>
```

```
show wrr-queue cos-map
```

- Displays the mapping of the CoS priority queues.

```
switch> show wrr-queue cos-map
CoS Value      : 0 1 2 3 4 5 6 7
Priority Queue : 1 1 2 2 3 3 4 4
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-23

The **show wrr-queue bandwidth** command is used to display the WRR bandwidth allocation for the four CoS priority queues.

The **show wrr-queue cos-map** command is used to display the mapping of the CoS priority queues.

Monitoring Queuing on Catalyst 2950 Switches (Cont.)

```
switch>
```

```
show mls qos interface [interface-id] [policers]
```

- Displays QoS information at the interface level.

```
switch> show mls qos interface fastethernet0/1
FastEthernet0/1
trust state:trust cos
trust mode:trust cos
COS override:dis
default COS:0
pass-through:none
trust device:cisco-phone
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3-5-24

Use the **show mls qos interface** command to display QoS information at the interface level. Although it will be visible in command-line interface (CLI) help strings, the **policers** keyword is available only when the Catalyst 2950 switch is running the enhanced software image.

The syntax for the **show mls qos interface** command is as follows:

```
show mls qos interface [interface-id] [policers]
```

Syntax Description

Parameter	Description
<i>interface-id</i>	(Optional) Displays QoS information for the specified interface.
policers	(Optional) Displays all the policers configured on the interface, their settings, and the number of policers unassigned. Available only when the switch is running the EI software.

Monitoring Queuing on Catalyst 2960 Switches

switch>

```
show mls qos interface [interface-id] [buffers | queueing  
| statistics] [ | {begin | exclude | include} expression]
```

- Displays QoS information at the interface level.

```
switch> show mls qos interface gigabitethernet0/2 buffers  
GigabitEthernet0/2  
The port is mapped to qset : 1  
The allocations between the queues are : 25 25 25 25
```

- Displays the buffer allocation among the queues.

```
switch> show mls qos interface gigabitethernet0/2 queueing  
GigabitEthernet0/2  
Egress Priority Queue :enabled  
Shaped queue weights (absolute) : 25 0 0 0  
Shared queue weights : 25 25 25 25  
The port bandwidth limit : 100 (Operational Bandwidth:100.0)  
The port is mapped to qset : 1
```

- Displays the queuing strategy and the weights corresponding to the queues.
- The egress expedite queue overrides the configured SRR weights.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-25

To display QoS information at the interface level, use the **show mls qos interface** command.
To display the buffer allocation among the queues, use the **buffer** keyword with this command.
To display the queuing strategy and the weights corresponding to the queues, use the **queueing** keyword with this command.

You can use the **show running-config** command to determine whether QoS is enabled and whether the egress expedite queue is enabled or disabled.

The syntax for the **show mls qos interface** command on Catalyst 2960 Series switches running Cisco IOS Software Release 12.2(46)SE is as follows:

```
show mls qos interface [interface-id] [buffers | queueing |  
statistics] [ | {begin | exclude | include} expression]
```

Syntax Description

Parameter	Description
<i>interface-id</i>	(Optional) Displays QoS information for the specified port. Valid interfaces include physical ports.
buffers	(Optional) Displays the buffer allocation among the queues.
queueing	(Optional) Displays the queueing strategy (shared or shaped) and the weights corresponding to the queues.
statistics	(Optional) Displays statistics for sent and received Differentiated Services Code Points (DSCPs) and class of service (CoS) values, the number of packets enqueued or dropped per egress queue, and the number of in-profile and out-of-profile packets for each policer.
begin	(Optional) Display begins with the line that matches the <i>expression</i> .
exclude	(Optional) Display excludes lines that match the <i>expression</i> .
include	(Optional) Display includes lines that match the specified <i>expression</i> .
<i>expression</i>	Expression in the output to use as a reference point.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- The number and capabilities of queues on Catalyst switches depend upon the model of the switch, supervisor, and line cards.
- With WRR, which prevents the low-priority queues from being neglected during periods of high-priority traffic, different weights are assigned to each queue. On most Catalyst switches, a single priority queue can be configured with WRR to ensure priority dispatch of voice traffic.
- Use the following commands to configure WRR with PQ on a Catalyst 2950 switch:
 - **wrr-queue cos-map**: To configure CoS-to-queue mappings for PQ
 - **wrr-queue bandwidth**: To assign weights to the four CoS priority queues
- WTD is used as the congestion-avoidance mechanism and SRR is used as the scheduling service on Catalyst 2960 Series switches. WTD uses the QoS label that is assigned to a frame to subject the frame to a threshold; if the threshold is exceeded for that QoS label, the switch drops the frame.

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6-26

Summary (Cont.)

- Use the following commands to configure WTD on a Catalyst 2960 switch:
 - **mls qos srr-queue input cos-map** and **mls qos srr-queue output cos-map**: To map CoS values to queues and threshold IDs
 - **mls qos srr-queue input threshold**: To assign threshold percentages to **ingress** queues
 - **mls qos queue-set output threshold**: To guarantee availability of buffers, set thresholds, configure maximum allocation for **egress** queue sets
 - **mls qos queue-set output buffers**: To allocate buffers to **egress** queue-sets
 - **queue-set**: To map ports to queue sets
- SRR, which controls the rate at which packets are sent, has two modes:
 - Shaped: Queues are guaranteed a percentage of bandwidth; queues cannot utilize unused bandwidth allocated to other queues.
 - Shared: Queues share bandwidth according to configured weights; bandwidth is guaranteed; queues can use unused bandwidth allocated to other queues.

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6-27

Summary (Cont.)

- Use the following commands to configure SRR on a Catalyst 2960 switch:
 - **mls qos srr-queue input priority-queue**: To configure the **ingress** priority queue and guarantee bandwidth on the internal ring if the ring is congested.
 - **mls qos srr-queue input bandwidth**: To assign SRR weights to an **ingress** queue.
 - **srr-queue bandwidth share**: To assign shared weights and enable bandwidth sharing on the four **egress** queues mapped to a port.
 - **srr-queue bandwidth shape**: To assign the shaped weights and enable bandwidth shaping on the four **egress** queues mapped to a port.
 - **priority-queue out**: To enable the **egress** expedite queue on a port.
- The **show mls qos maps** command is used to display QoS mapping information on Catalyst 2950 and 2960 switches.

Module Summary

This topic summarizes the key points discussed in this module.

Module Summary

- Congestion can occur at any point in the network, but particularly at points of speed mismatches and traffic aggregation. Queuing algorithms such as **FIFO**, **priority**, and **round robin** are used to manage congestion.
- Each physical interface has a hardware and a software queuing system.
- **WFQ** was developed to overcome the limitations of the more basic queuing methods.
- **CBWFQ** extends the standard WFQ functionality to provide support for user-defined traffic classes. **LLQ** extends the functionality of CBWFQ by adding priority queues for time-sensitive traffic such as voice and video.
- **PQ**, **WRR**, **WRR with a PQ**, and **SRR** are queuing methods used for Catalyst switches.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-5.1

Effective congestion management is key to QoS in converged networks. Low-latency traffic such as voice and video must be constantly moved to high-priority queues in order to ensure reasonable quality.

Cisco routers offer a variety of simple (FIFO, PQ, and CQ) and sophisticated (WFQ, CBWFQ, and LLQ) queuing algorithms to provide effective congestion management on converged networks. LLQ, the most sophisticated queuing algorithm, was specifically designed to provide the highest QoS to voice traffic.

Cisco switches offer a variety of queuing capabilities, depending upon the model of switch being used. Queuing methods available for use on Catalyst switches include PQ, WRR, WRR with a priority queue, and SRR.

References

For additional information, refer to these resources:

- To learn more about congestion and queuing, refer to *Understanding Delay in Packet Voice Networks* at this URL:
http://www.cisco.com/en/US/tech/tk652/tk698/technologies_white_paper09186a00800a8993.shtml
- To learn more about congestion and queuing, refer to *Understanding Jitter in Packet Voice Networks (Cisco IOS Platforms)* at this URL:
http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a00800945df.shtml
- To learn more about configuring WFQ, refer to *Configuring Weighted Fair Queueing* at this URL: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/config_wfq.html
- To learn more about queuing on the Cisco Catalyst 3550 Series switches, refer to *Configuring QoS* at this URL:
http://www.cisco.com/en/US/docs/switches/lan/catalyst3550/software/release/12.2_44_se/configuration/guide/swqos.html
- To learn more about queuing on the Cisco Catalyst 2950 Series switches, refer to *Configuring QoS* at this URL:
http://www.cisco.com/en/US/docs/switches/lan/catalyst2950/software/release/12.1_22_ea5/configuration/guide/swqos.html
- To learn more about queuing on the Cisco Catalyst 2960 Series switches, refer to *Configuring QoS* at this URL:
http://www.cisco.com/en/US/docs/switches/lan/catalyst2960/software/release/12.2_50_se/configuration/guide/swqos.html
- To learn more about queuing on the Cisco Catalyst 4500 Series switches, refer to *Configuring QoS* at this URL:
<http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.1/12ew/configuration/guide/qos.html>
- To learn more about queuing on the Cisco Catalyst 6000 Series switches, refer to *Configuring QoS* at this URL:
http://www.cisco.com/en/US/tech/tk543/tk762/technologies_white_paper09186a00800b0828.shtml

Module Self-Check

Use the questions here to review what you learned in this module. The correct answers and solutions are found in the Module Self-Check Answer Key.

- Q1) What happens when the highest-priority queue becomes congested in a priority queuing algorithm? (Source: Introducing Queuing)
- A) All the other queues starve.
 - B) Tail dropping focuses on the highest-priority queue.
 - C) Other queues are served on a round-robin basis.
 - D) Packets in the highest-priority queue are moved to a lower-priority queue.
- Q2) In WRR implementation, using a byte threshold as a measurement of each queue share of bandwidth, given an MTU of 2000 and a byte-count of 4000, what would the router do with the next packet for queue 2 (800 bytes) if the router had just dispatched two packets from queue 2 (sizes 2000 and 1600) to the hardware queue? (Source: Introducing Queuing)
- A) tail drop the next packet
 - B) dispatch the first packet from the next queue
 - C) dispatch the next packet to the hardware queue
 - D) split the packet and transmit the first 400 bytes
- Q3) Given that the hardware queue is *not* full, how will the next packet be serviced by the software queue? (Source: Understanding Queuing Implementations)
- A) software queue will be bypassed
 - B) software queue will enqueue the packet
 - C) software queue will expedite the packet
 - D) software queue will only meter the packet
- Q4) How does WFQ implement tail dropping? (Source: Configuring FIFO and WFQ)
- A) drops the last packet to arrive
 - B) drops all nonvoice packets first
 - C) drops the lowest-priority packets first
 - D) drops packets from the most aggressive flows
- Q5) Consider that a WFQ system has a modest hold-queue limit of ten ($HQO = 10$) and a congestive discard threshold of eight ($CDT = 8$), and that there are already eight packets in the system. If a newly arriving packet had the worst finish time of all packets in the system, what would happen to the packet? (Source: Configuring FIFO and WFQ)
- A) It would be dropped.
 - B) It would be enqueued.
 - C) It would be buffered until a spot in a queue came open.
 - D) It would be dispatched.
- Q6) Which of the following is the default dropping scheme for CBWFQ? (Source: Configuring CBWFQ and LLQ)
- A) RED
 - B) WRED
 - C) tail drop
 - D) class-based policing

- Q7) What does LLQ bring to CBWFQ? (Source: Configuring CBWFQ and LLQ)
- A) strict priority scheduling
 - B) alternate priority scheduling
 - C) non-policed queues for low-latency traffic
 - D) special voice traffic classification and dispatch
- Q8) Which type of traffic should you limit the use of the **priority** command to? (Source: Configuring CBWFQ and LLQ)
- A) critical data traffic
 - B) voice traffic
 - C) bursty traffic
 - D) video and teleconferencing, available (ABR) traffic
- Q9) When WRR with an expedite queue has been configured on a Cisco Catalyst 2950 switch, which queue is emptied before any other queues are serviced? (Source: Configuring LAN Congestion Management)
- A) queue 1
 - B) queue 2
 - C) queue 3
 - D) queue 4

Module Self-Check Answer Key

Q1)	A
Q2)	C
Q3)	A
Q4)	D
Q5)	A
Q6)	C
Q7)	A
Q8)	B
Q9)	D

Congestion Avoidance

Overview

Congestion is a normal occurrence in networks. Whether congestion occurs as a result of a lack of buffer space, network aggregation points, or a low-speed wide-area link, many congestion management techniques exist to ensure that specific applications and traffic classes are given their share of available bandwidth when congestion occurs. Congestion management does, however, come at a price. When congestion occurs, some traffic is delayed or even dropped at the expense of other traffic. When drops occur, different problems may arise which can exacerbate the congestion, such as retransmissions and TCP global synchronization in TCP/IP networks. Network administrators can use congestion avoidance mechanisms to reduce the negative effects of congestion by penalizing the most aggressive traffic streams as software queues begin to fill. This module describes the problems with TCP congestion management and the benefits of deploying congestion avoidance mechanisms in a network.

Module Objectives

Upon completing this module, you will be able to use Cisco quality of service (QoS) congestion avoidance mechanisms to reduce the effects of congestion on the network. This ability includes being able to meet these objectives:

- Explain the problems that may result from the limitations of TCP congestion management mechanisms on a converged network
- Explain how RED can be used to avoid congestion
- Configure CBWRED to avoid congestion
- Configure ECN to enhance the congestion avoidance features of WRED

Introducing Congestion Avoidance

Overview

TCP supports traffic management mechanisms such as slow start and fast retransmit. When congestion occurs, tail-dropping the TCP traffic can cause TCP global synchronization, resulting in poor bandwidth use. This lesson describes how TCP manages the traffic flow between two hosts, and the effects of tail-dropping on TCP traffic.

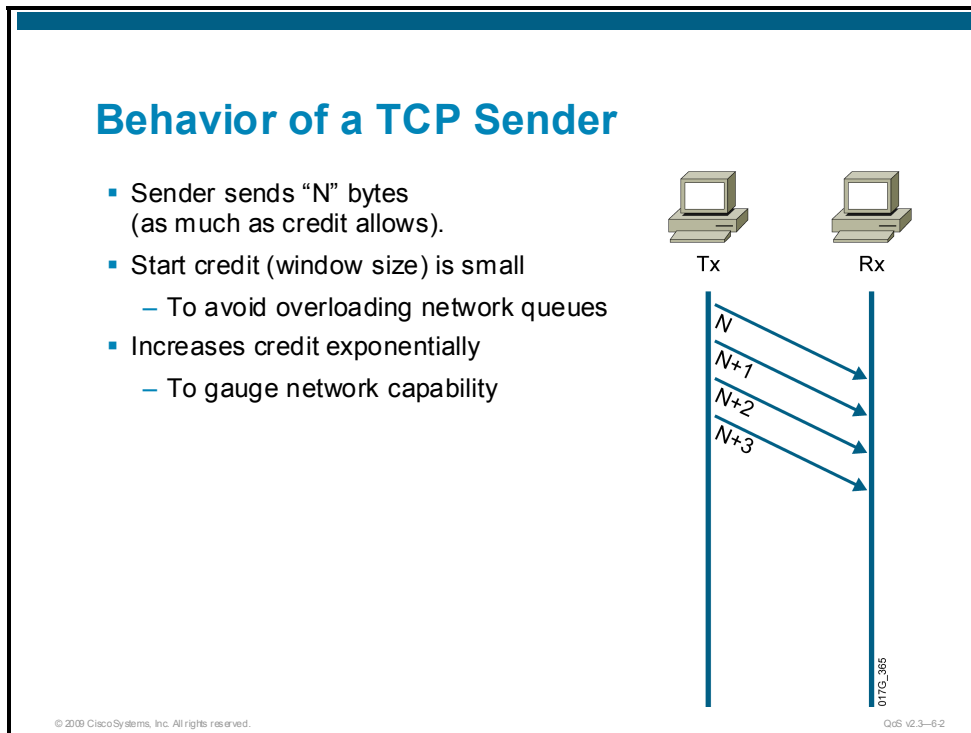
Objectives

Upon completing this lesson, you will be able to explain the problems that may result from the limitations of TCP congestion management mechanisms on a converged network. This ability includes being able to meet these objectives:

- Explain the behavior of TCP senders and receivers when sending packets
- Explain how TCP responds to congestion
- Describe the default mechanism for managing interface congestion with tail drop
- Describe the limitations of using tail drop as a congestion management mechanism

Behavior of TCP Senders and Receivers

This topic describes the behavior of TCP senders and receivers when sending packets.



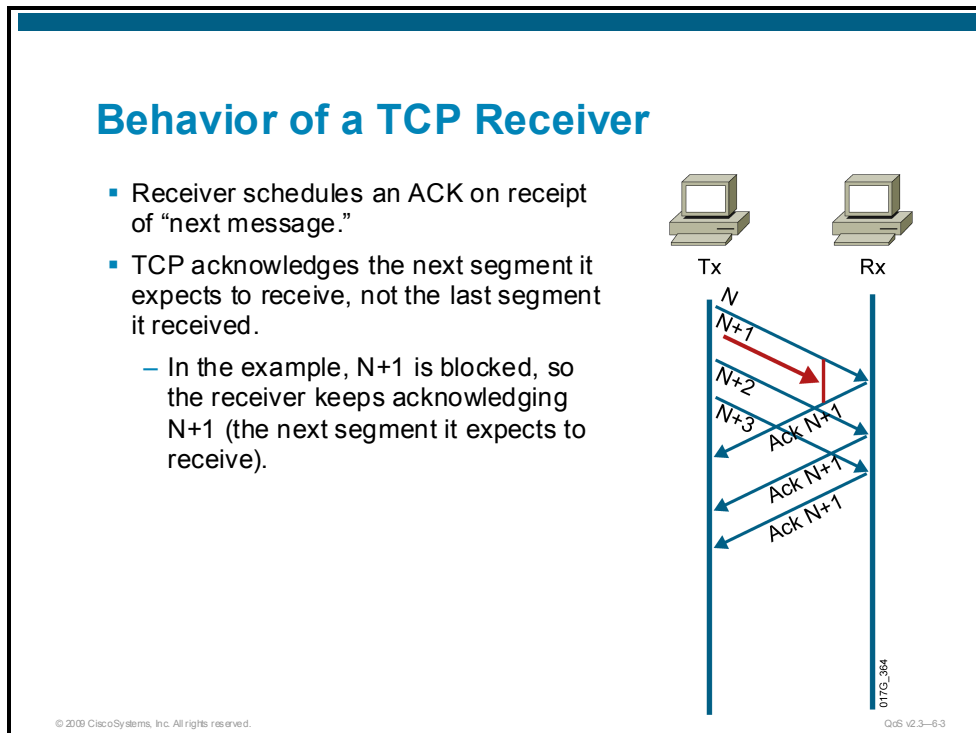
Before any data is transmitted using TCP, a connection must first be established between the transmitting and receiving hosts. When the connection is initially established, the two hosts must agree on certain parameters that will be used during the communication session. One of the parameters that must be decided is called the window size, or how many data bytes to transmit at a time. Initially, TCP sends a small number of data bytes, and then exponentially increases the number sent. For example, a TCP session originating from host A begins with a window size of 1 and therefore sends one packet. When host A receives a positive ACK from the receiver, Host A increases its window size to 2. Host A then sends 2 packets, receives a positive ACK, and increases its window size to 4, and so on.

Note TCP tracks window size by byte count. For purposes of illustration packets, N is used.

In traditional TCP, the maximum window size is 64 KB (65,535 bytes). Extensions to TCP, specified in RFC 1323, allow for tuning TCP by extending the maximum TCP window size to 2^{30} bytes. TCP extensions for high performance, although supported on most operating systems, may not be supported on your system.

Example: Windowing in TCP

After connecting to a website, a file transfer using the FTP download is initiated. Watching the progress of the transfer, the bytes-per-second counter steadily increases during the file transfer. This is an example of TCP windowing in action.



When the receiver receives a data segment, the receiver checks that data segment sequence number (byte count). If the data received fills in the next sequence of numbers expected, the receiver indicates that the data segment was received in order. The receiver then delivers all the data that it holds to the target application, and updates the sequence number to reflect the next byte number in expected order.

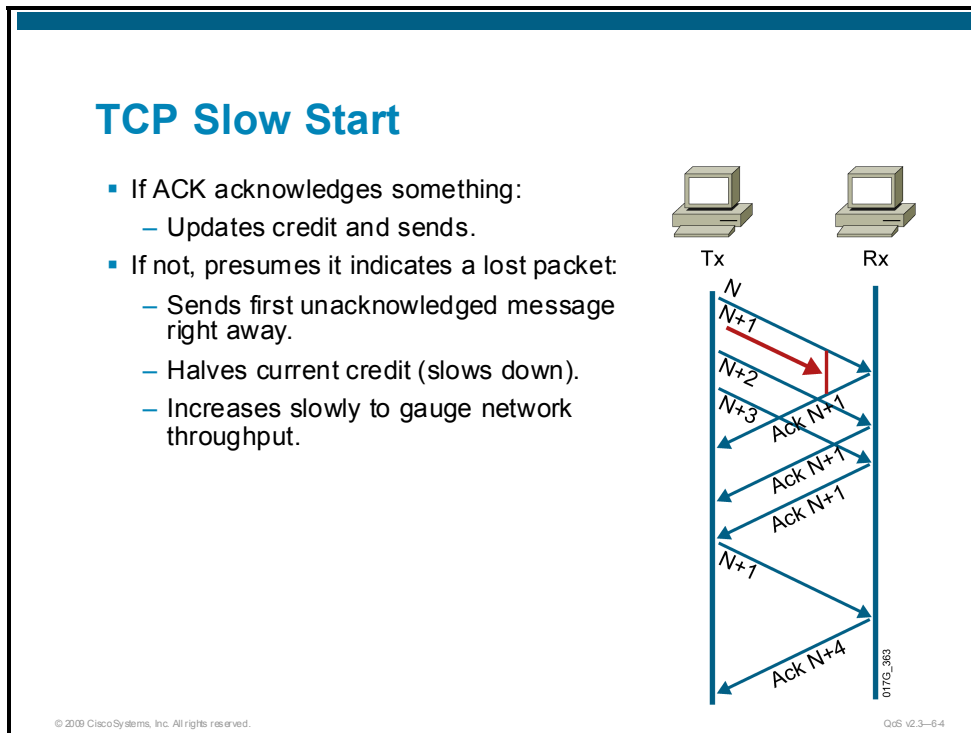
When this process is complete, the receiver performs one of these actions:

- Immediately transmits an acknowledgment (ACK) to the sender
- Schedules an ACK to be transmitted to the sender after a short delay

The ACK notifies the sender that the receiver received all data segments up to but not including the byte number in the new sequence number. Receivers usually try to send an ACK in response to alternating data segments they receive. They send the ACK because, for many applications, if the receiver waits out a small delay, it can efficiently piggyback its reply acknowledgment on a normal response to the sender. However, when the receiver receives a data segment out of order, it immediately responds with an ACK to direct the sender to retransmit the lost data segment.

Congestion and TCP

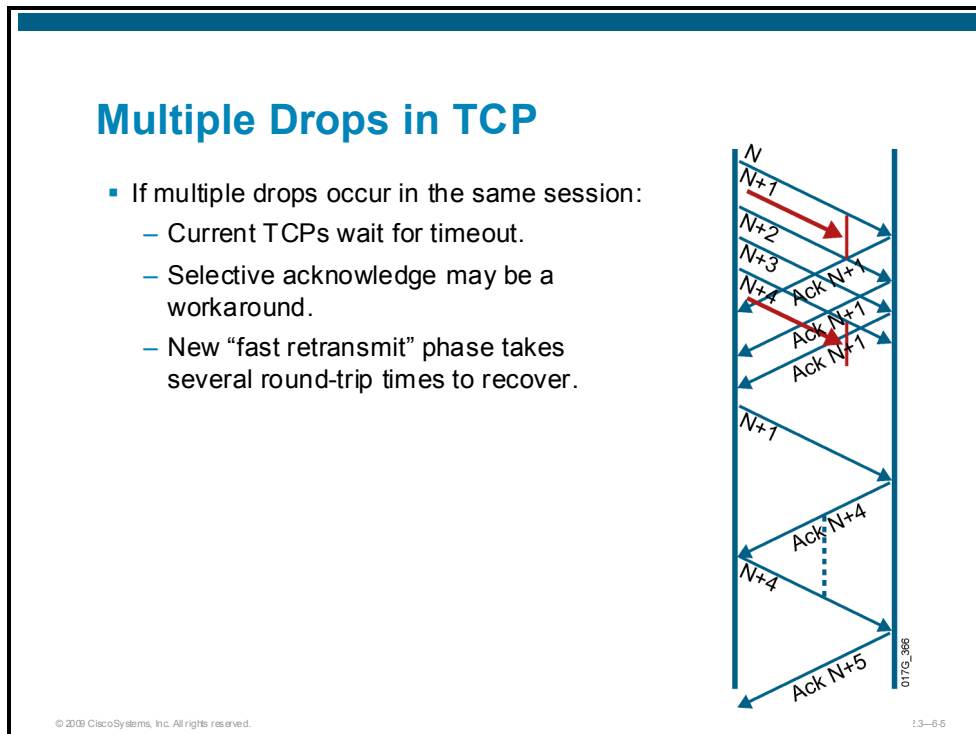
This topic describes the TCP response to congestion.



When the sender receives an ACK, the sender determines if any data is outstanding:

- If no data is outstanding, the sender determines that the ACK is a keepalive, meant to keep the line active, and it does nothing.
- If data is outstanding, the sender determines whether the ACK indicates that the receiver has received some or none of the data.
 - If the ACK acknowledges receipt of some data sent, the sender determines if new credit has been granted to allow it to send more data.
 - When the ACK acknowledges receipt of none of the sent data and there is outstanding data, the sender interprets the ACK to be a repeatedly sent ACK. This condition indicates that some data was received out of order, forcing the receiver to retransmit the first ACK, and that a second data segment was received out of order, forcing the receiver to retransmit the second ACK. In most cases, the receiver would receive two segments out of order, because one of the data segments had been dropped.

When a TCP sender detects a dropped data segment, it retransmits the segment. Then the sender slows its transmission rate so that the rate is half of what it was before the drop was detected. This is known as the TCP slow-start mechanism. In the figure, a station transmits three packets to the receiving station. Unfortunately, the first packet is dropped somewhere in the network. Therefore the receiver sends an ACK 1, to request the missing packet. Because the transmitter does not know if the ACK was just a duplicate ACK, it will wait for three ACK 1 packets from the receiver. Upon receipt of the third ACK, the missing packet, packet 1, is resent to the receiver. The receiver now sends an ACK 4 indicating that it has already received packets 2 and 3 and is ready for the next packet.



Although the TCP slow-start behavior is appropriately responsive to congestion, problems can arise when multiple TCP sessions are concurrently carried on the same router and all TCP senders slow down transmission of packets at the same time.

If a TCP sender does not receive acknowledgement for sent segments, it cannot wait indefinitely before it assumes that the data segment that was sent never arrived at the receiver. TCP senders maintain the retransmission timer to trigger a segment retransmission. The retransmission timer can impact TCP performance. If the retransmission timer is too short, duplicate data will be sent into the network unnecessarily. If the retransmission timer is too long, the sender will wait (remain idle) for too long, slowing down the flow of data.

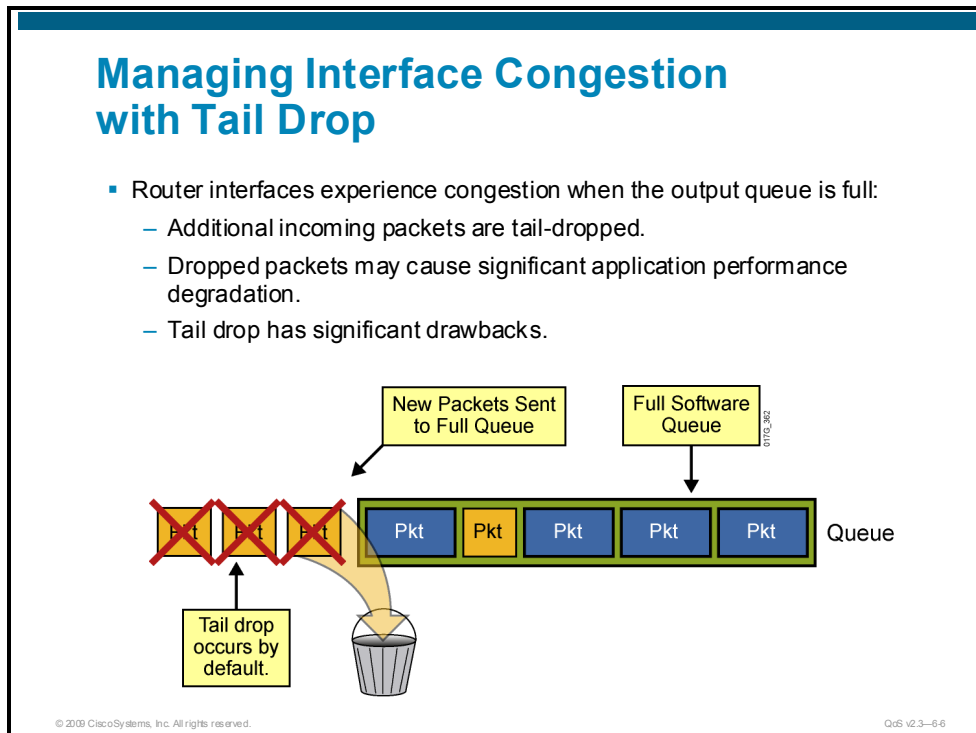
The selective acknowledgment (SACK) mechanism, as proposed in RFC 2018, can improve the time it takes for the sender to recover from multiple packet losses, because noncontiguous blocks of data can be acknowledged, and the sender only has to retransmit data that is actually lost. SACK is used to convey extended acknowledgement information from the receiver to the sender to inform the sender of noncontiguous blocks of data that have been received. Using the example in the slide, instead of sending back an ACK $N + 1$, the receiver can send a SACK $N + 1$ and also indicate back to the sender that $N + 3$ has been correctly received with the SACK option.

In standard TCP implementations, a TCP sender can only discover that a single packet has been lost each round-trip time (RTT), causing poor TCP performance when multiple packets are lost.

Remember, the sender must receive three duplicate ACK packets before it realizes that a packet has been lost. As a result of receiving the third ACK, the sender will immediately send the segment referred to by the ACK. This TCP behavior is called fast retransmit.

Managing Interface Congestion with Tail Drop

This topic describes the default mechanism for managing interface congestion with tail drop.



When an interface on a router cannot transmit a packet immediately, the packet is queued, either in an interface transmit (Tx) ring, or the interface output hold queue, depending on the switching path that is used. Packets are then taken out of the queue and eventually transmitted on the interface.

If the arrival rate of packets to the output interface exceeds the router capability to buffer and forward traffic, the queues increase to their maximum length and the interface becomes congested. Tail drop is the default queuing response to congestion. Tail drop treats all traffic equally and does not differentiate between classes of service. Applications may suffer performance degradation due to packet loss caused by tail drop. When the output queue is full and tail drop is in effect, all packets trying to enter (at the tail of) the queue are dropped until the congestion is eliminated and the queue is no longer full.

Weighted fair queuing (WFQ), if configured on an interface, has a more elaborate scheme for dropping traffic. It is able to punish the most aggressive flows via its congestive discard threshold (CDT)-based dropping algorithm. Unfortunately, WFQ does not scale to backbone speeds.

Tail Drop Limitations

This topic describes the limitations of using tail drop as a congestion management mechanism.

Tail Drop Limitations

- Tail drop should be avoided because it contains significant flaws:
 - TCP synchronization
 - TCP starvation
 - No differentiated drop

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.7

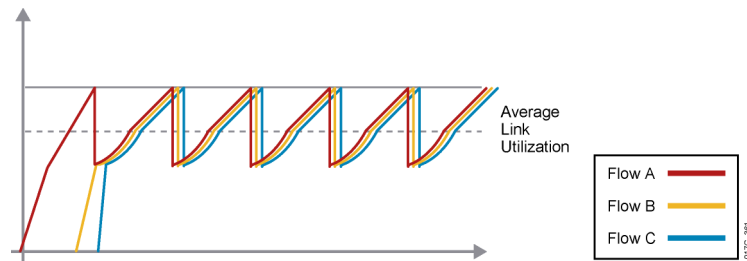
The simple tail-drop scheme does not work well in environments with a large number of TCP flows or in environments in which selective dropping is required. Understanding the network interaction between TCP stack intelligence and dropping is required to implement a more efficient and fair dropping scheme, especially in service provider environments.

Tail drop has the following shortcomings:

- When congestion occurs, dropping affects most of the TCP sessions, which simultaneously back off and then restart again. This causes inefficient link utilization at the congestion point (TCP global synchronization).
- TCP starvation, in which all buffers are temporarily seized by aggressive flows, and normal TCP flows experience buffer starvation.
- There is no differentiated drop mechanism, and therefore premium traffic is dropped in the same way as best-effort traffic.

TCP Synchronization

- Multiple TCP sessions start at different times.
- TCP window sizes are increased.
- Tail drops cause many packets of many sessions to be dropped at the same time.
- TCP sessions restart at the same time (synchronized).



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.8

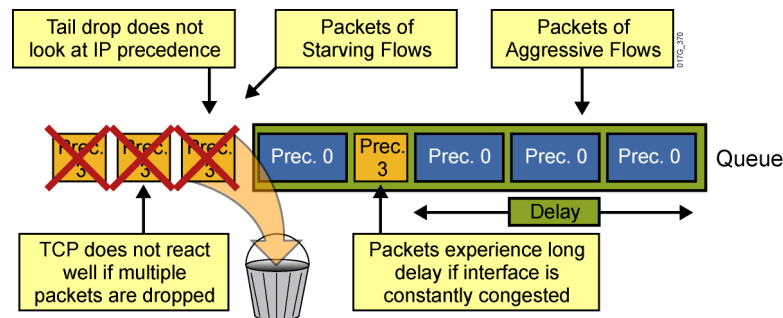
A router can handle multiple concurrent TCP sessions. It is likely that when traffic exceeds the queue limit, it exceeds this limit due to the bursty nature of packet networks. However, there is also a high probability that excessive traffic depth caused by packet bursts are temporary and that traffic does not stay excessively deep except at points where traffic flows merge, or at edge routers.

If the receiving router drops all traffic that exceeds the queue limit, as is done with tail drop by default, many TCP sessions simultaneously go into slow start. Consequently, traffic temporarily slows down to the extreme and then all flows slow-start again. This activity creates a condition called global synchronization.

Global synchronization occurs as waves of congestion crest only to be followed by troughs during which the transmission link is not fully used. Global synchronization of TCP hosts can occur because packets are dropped all at once. Global synchronization occurs when multiple TCP hosts reduce their transmission rates in response to packet dropping. When congestion is reduced, their transmission rates are increased. The most important point is that the waves of transmission known as global synchronization result in significant link underutilization.

TCP Delay, Jitter, and Starvation

- Constant high buffer usage (long queue) causes **delay**.
- More aggressive flows can cause other flows to **starve**.
- No differentiated dropping occurs.



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.9

During periods of congestion, packets are queued up to the full queue length, which also causes increased delay for packets that are already in the queue. In addition, queuing introduces unequal delays for packets of the same flow, thus producing jitter.

Another TCP-related phenomenon that reduces optimal throughput of network applications is TCP starvation. When multiple flows are established over a router, some of these flows may be much more aggressive than other flows. For instance, when a file transfer application TCP transmit window increases, the TCP session can send a number of large packets to its destination. The packets immediately fill the queue on the router, and other, less aggressive flows can be starved because there is no differentiated treatment indicating which packets should be dropped. As a result, these less aggressive flows are tail-dropped at the output interface.

Based on the knowledge of TCP behavior during periods of congestion, you can conclude that tail drop is not the optimal mechanism for congestion avoidance and therefore should not be used. Instead, more intelligent congestion avoidance mechanisms should be used that slow down traffic before actual congestion occurs.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- TCP sender sends as many bytes as credit allows. TCP receiver acknowledges the next segment it expects to receive.
- TCP uses windowing and the TCP slow-start mechanism as its means of controlling congestion.
- By default, routers resort to tail drop, relying on TCP congestion controls when queues become full.
- Tail drop causes significant issues including TCP synchronization, starvation, and delay. TCP synchronization decreases the average utilization of network links.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--6-10

Introducing RED

Overview

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottleneck points. Congestion avoidance is achieved through packet dropping using a more complex dropping technique than simple tail drop.

This lesson describes the congestion avoidance technique random early detection (RED) and its scalable dropping method, which is suitable for low- and high-speed networks.

Objectives

Upon completing this lesson, you will be able to explain how RED can be used to avoid congestion. This ability includes being able to meet these objectives:

- Describe RED and how it can be used to prevent congestion
- Describe the elements of a RED traffic profile used to implement the RED packet-dropping strategy
- Describe the different packet drop modes of RED
- Describe the effects of using RED on TCP traffic by comparing TCP traffic flows both before and after the application of RED
- Identify the points in a network where congestion avoidance mechanisms are commonly deployed in enterprise and service provider networks

Random Early Detection

This topic describes RED and how it can be used to prevent congestion.

Random Early Detection

- Tail drop can be avoided if congestion is prevented.
- RED is a mechanism that randomly drops packets before a queue is full.
- RED increases drop rate as the average queue size increases.
- RED result:
 - TCP sessions slow down to the approximate rate of output-link bandwidth.
 - Average queue size is small (much less than the maximum queue size).
 - TCP sessions are desynchronized by random drops.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6.2

RED is a dropping mechanism that randomly drops packets before a queue is full. The dropping strategy is based primarily on the average queue length—that is, when the average size of the queue increases, RED will be more likely to drop an incoming packet than when the average queue length is shorter.

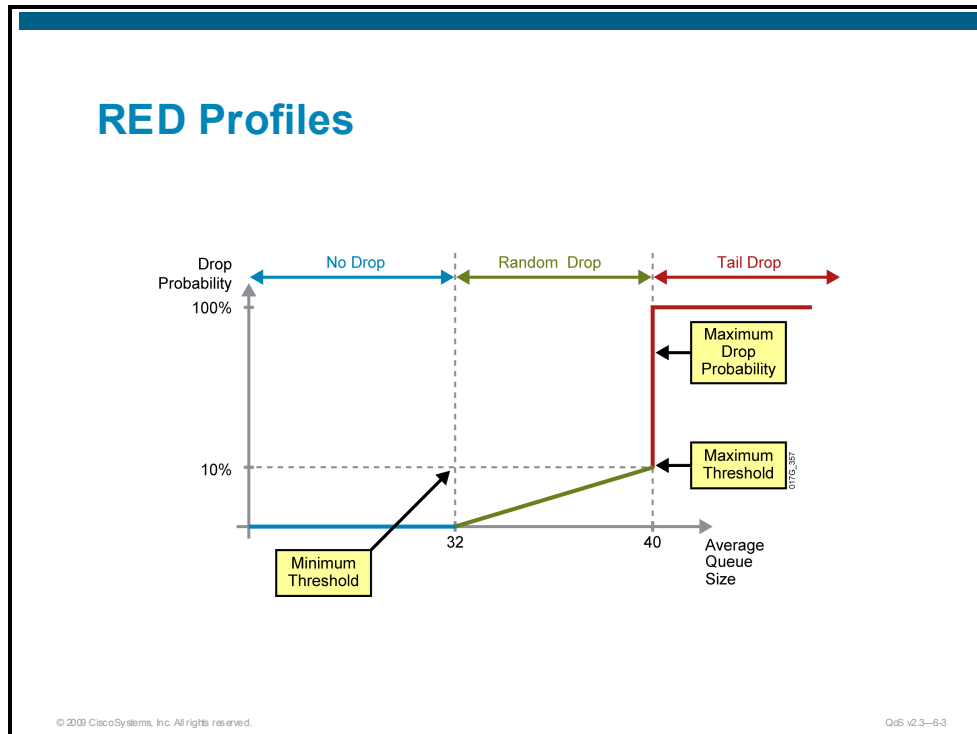
Because RED drops packets randomly, it has no per-flow intelligence. The rationale is that an aggressive flow will represent most of the arriving traffic and therefore it is likely that RED will drop a packet of an aggressive session. RED therefore punishes more aggressive sessions with higher statistical probability and is, therefore, able to somewhat selectively slow down the most significant cause of congestion. Directing one TCP session at a time to slow down allows for full utilization of the bandwidth, rather than utilization that manifests itself as crests and troughs of traffic.

As a result of implementing RED, the problem of TCP global synchronization is much less likely to occur, and TCP can utilize link bandwidth more efficiently. In RED implementations, the average queue size also decreases significantly, as the possibility of the queue filling up is reduced. This is because of very aggressive dropping in the event of traffic bursts, when the queue is already quite full.

RED distributes losses over time and normally maintains a low queue depth while absorbing traffic spikes. RED can also utilize IP precedence or differentiated services code points (DSCPs) to establish different drop profiles for different classes of traffic, in which case it is referred to as weighted random early detect (WRED) or DSCP-based WRED, respectively.

RED Profiles

This topic describes the elements of a RED traffic profile used to implement the RED packet-dropping strategy.



A RED traffic profile is used to determine the packet-dropping strategy and is based on the average queue length. The probability of a packet being dropped is based on three configurable parameters contained within the RED profile:

- **Minimum threshold:** When the average queue length is equal or above the minimum threshold, RED starts dropping packets. The rate of packet drop increases linearly as the average queue size increases, until the average queue size reaches the maximum threshold.
- **Maximum threshold:** When the average queue size is above the maximum threshold, all packets are dropped.
- **Mark probability denominator:** This is the fraction of packets that are dropped when the average queue depth is at the maximum threshold. For example, if the denominator is 512, one out of every 512 packets is dropped when the average queue is at the maximum threshold. The linear increase of packet drops from the minimum threshold (0 drops) to the maximum threshold is based on this parameter and the queue size between the minimum and maximum thresholds.

The minimum threshold value should be set high enough to maximize the link utilization. If the minimum threshold is too low, packets may be dropped unnecessarily, and the transmission link will not be fully used.

The difference between the maximum threshold and the minimum threshold should be large enough to avoid global synchronization. If the difference is too small, many packets may be dropped at once, resulting in global synchronization.

The mark probability has the effect of controlling the number of packets that are dropped when the average queue length reaches the maximum threshold. If the value is set too low, it will result in too many dropped packets. If the value is set too large, RED dropping can be rendered ineffective.

RED Modes

This topic describes the different packet-drop modes of RED.

RED Modes

- RED has three modes:
 - **No drop**: When the average queue size is between 0 and the minimum threshold.
 - **Random drop**: When the average queue size is between the minimum and the maximum threshold.
 - **Full drop (tail drop)**: When the average queue size is at maximum threshold or above.
- Random drop should prevent congestion (prevent tail drops).

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.4

Based on the average queue size, RED has three dropping modes:

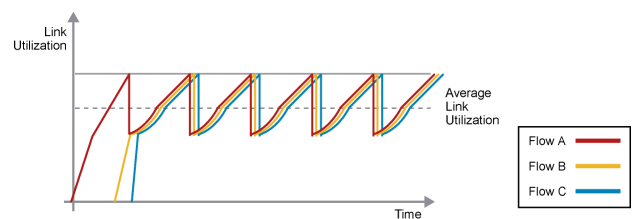
- When the average queue size is between 0 and the configured minimum threshold, no drops occur and all packets are queued.
- When the average queue size is between the configured minimum threshold and the configured maximum threshold, random drops occur, which is linearly proportional to the mark probability denominator and the average queue length.
- When the average queue size is at or higher than the maximum threshold, RED performs full (tail) drop in the queue. This is unlikely, as RED should slow down TCP traffic ahead of congestion. If a lot of non-TCP traffic is present, RED cannot effectively drop traffic to reduce congestion, and tail drops are likely to occur.

TCP Traffic Before and After RED

This topic describes the effects of using RED on TCP traffic by comparing TCP traffic flows both before and after the application of RED.

TCP Traffic Before RED

- **TCP synchronization** prevents average link utilization close to the link bandwidth.
- **Tail drops** cause TCP sessions to go into slow-start.

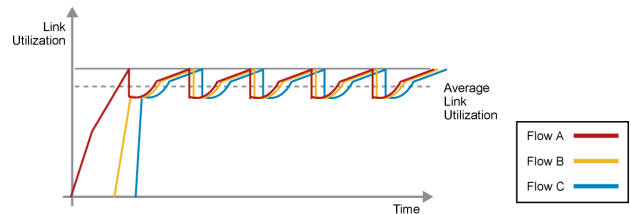


The figure shows TCP throughput behavior compared to link bandwidth in a congested network scenario where the tail-drop mechanism is in use on a router. The global synchronization phenomenon causes all sessions to slow down when congestion occurs. All sessions are penalized when tail drop is used because it drops packets with no discrimination between individual flows.

When all sessions slow down, congestion on the router interface is removed and all TCP sessions restart their transmission at roughly the same time. Again, the router interface quickly becomes congested, causing tail drop. As a result, all TCP sessions back off again. This behavior cycles constantly, resulting in a link that is generally underutilized.

TCP Traffic After RED

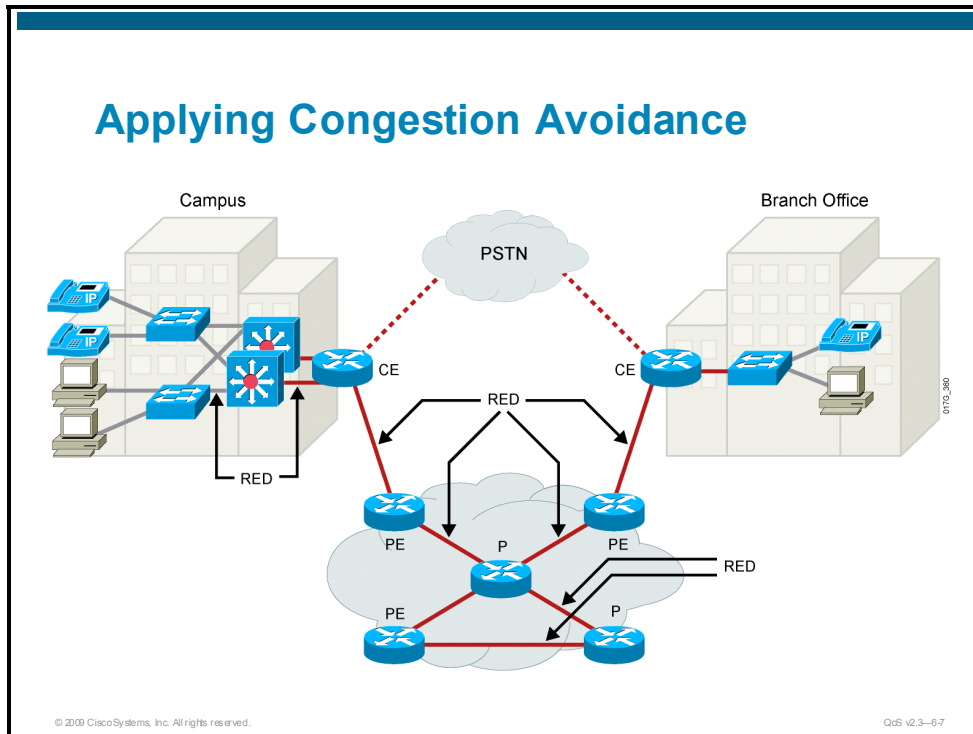
- Average link utilization is much closer to link bandwidth.
- **Random drops** cause TCP sessions to reduce window sizes.



This figure shows TCP throughput behavior compared to link bandwidth in a congested network scenario in which RED has been configured on a router. RED randomly drops packets, influencing a small number of sessions at a time, before the interface reaches congestion. Overall throughput of sessions is increased, as well as average link utilization. Global synchronization is very unlikely to occur, due to selective, but random, dropping of adaptive traffic.

Applying Congestion Avoidance

This topic describes the points in a network where congestion avoidance mechanisms are commonly deployed in enterprise and service provider networks.



RED is most useful in enterprise and service provider networks on output interfaces where congestion is expected to occur. Edge routers or switches typically classify and mark packets as the packets enter the network. Congestion avoidance mechanisms can use these packet markings to indicate a set of drop criteria for a traffic stream.

Congestion avoidance mechanisms are also applicable to the campus or LAN environment. In these networks, congestion avoidance is best used on interfaces that connect to WAN gateways, because these interfaces are typically sites for congestion to occur.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- RED is a mechanism that randomly drops packets before a queue is full, preventing congestion and avoiding tail drop.
- RED operates by increasing the rate at which packets are dropped from queues as the average queue size increases.
- RED has three modes of operation: no drop, random drop, and full drop (tail drop).
- With RED, TCP global synchronization is eliminated and the average link utilization increases.
- RED is most useful in enterprise and service provider networks on output interfaces where congestion is expected to occur. Edge routers or switches typically classify and mark packets as they enter the network.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.8

Configuring Class-Based Weighted RED

Overview

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottleneck points using advanced packet discard techniques.

This lesson describes the weighted random early detection (WRED) congestion avoidance technique, which is the Cisco implementation of random early detection (RED).

Objectives

Upon completing this lesson, you will be able to configure class-based weighted random early detection (CBWRED) to avoid congestion. This ability includes being able to meet these objectives:

- Describe WRED and how it can be used to prevent congestion
- Describe the traffic profiles used in WRED implementations
- Identify the Cisco IOS commands required to configure CBWRED
- Identify the Cisco IOS commands required to configure DSCP-based CBWRED
- Identify the Cisco IOS commands used to monitor CBWRED

Weighted Random Early Detection

This topic describes WRED and how you can use it to prevent congestion.

Weighted Random Early Detection

- **WRED** can use multiple different RED profiles.
- Each profile is identified by:
 - **Minimum threshold**
 - **Maximum threshold**
 - **Maximum drop probability**
- WRED profile selection is based on:
 - **IP precedence** (8 profiles)
 - **DSCP** (64 profiles)
- WRED drops less important packets more aggressively than more important packets.
- WRED can be applied at the interface, VC, or class level.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.2

WRED combines RED with IP precedence or differentiated services code point (DSCP) and performs packet dropping based on IP precedence (WRED) or DSCP (DSCP-based WRED).

As with RED, WRED monitors the average queue length in the router and determines when to begin discarding packets based on the length of the interface queue. When the average queue length is greater than the user-specified minimum threshold, WRED begins to randomly drop packets (both TCP and User Datagram Protocol [UDP] packets) with a certain probability. If the average length of the queue continues to increase so that it becomes larger than the user-specified maximum threshold, WRED reverts to a tail-drop packet discard strategy, in which all incoming packets may be dropped.

The idea behind using WRED is to maintain the queue length at a level somewhere between the minimum and maximum thresholds, and to implement different drop policies for different classes of traffic. WRED can selectively discard lower-priority traffic when the interface becomes congested, and can provide differentiated performance characteristics for different classes of service. You can also configure WRED to achieve nonweighted RED behavior.

For interfaces configured to use the Resource Reservation Protocol (RSVP), WRED chooses packets from other flows to drop rather than the RSVP flows. Also, IP precedence or DSCP governs which packets are dropped, because traffic at a lower priority has a higher drop rate than traffic at a higher priority (and, therefore, lower priority is more likely to be slowed). In addition, WRED statistically drops more packets from large users than small users. Therefore, traffic sources that generate the most traffic are more likely to be slowed down than traffic sources that generate little traffic.

WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion. By dropping some packets early rather than waiting until the queue is full, WRED avoids dropping large numbers of packets at once and minimizes the chances of global synchronization. As a result, WRED maximizes the utilization of transmission lines.

WRED is only useful when the bulk of the traffic is TCP traffic. With TCP, dropped packets indicate congestion, so the packet source reduces its transmission rate. With other protocols, packet sources might not respond or might re-send dropped packets at the same rate, and so dropping packets might not decrease congestion.

WRED treats non-IP traffic as precedence 0, the lowest precedence. Therefore, non-IP traffic, in general, is more likely to be dropped than IP traffic.

WRED should be used wherever there is a potential bottleneck, or congested link—this could possibly be an access or edge link. However, WRED is normally used in the core routers of a network rather than at the network edge. Edge routers assign IP precedence or DSCP to packets as they enter the network. WRED uses these assigned values to determine how to treat different types of traffic.

WRED is not recommended for any voice queue, although WRED may be enabled on an interface carrying voice traffic. WRED will not throttle back voice traffic because voice traffic is UDP-based. The network itself should be designed not to lose voice packets because lost voice packets result in reduced voice quality. WRED controls congestion by impacting prioritized traffic other than voice, and avoiding congestion helps to ensure voice quality.

Class-Based WRED

- Class-based WRED is available when configured in combination with CBWFQ only.
- Using CBWFQ with WRED allows the implementation of DiffServ Assured Forwarding PHB.
- Class-based configuration of WRED is identical to standalone WRED.

© 2009 Cisco Systems, Inc. All rights reserved.

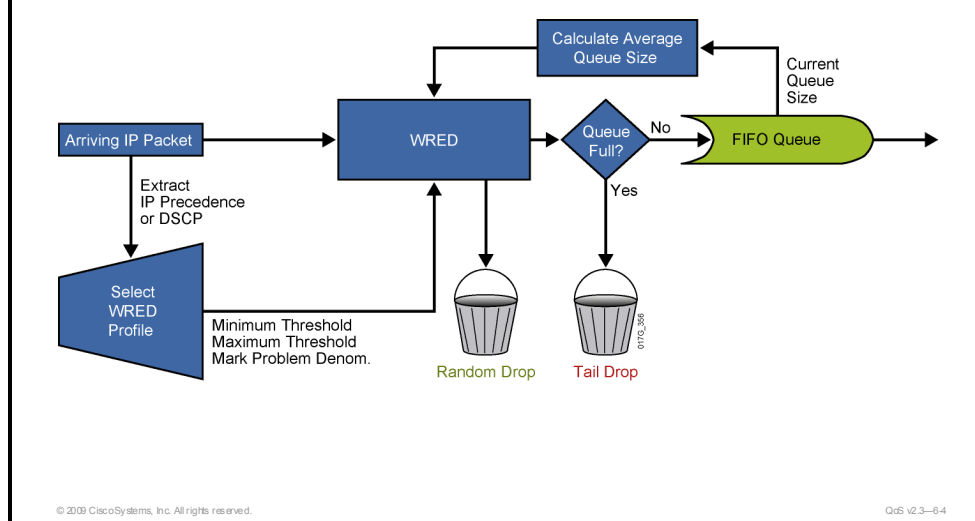
QoS v2.3-6.3

Congestion avoidance techniques monitor the network interface load in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through intelligent packet dropping techniques. Traditionally, Cisco IOS Software used standalone RED and WRED mechanisms to avoid congestion on an interface. Those mechanisms can perform a differentiated drop based on the IP precedence or DSCP value.

The class-based weighted fair queuing (CBWFQ) system supports the use of WRED inside the queuing system, thereby implementing CBWRED. Each class is queued in its separate queue and has a queue limit, performing tail drop by default. WRED can be configured as the preferred dropping method in a queue, implementing a differentiated drop based on traffic class, and further, on the IP precedence or DSCP value.

Note	The combination of CBWFQ and WRED on a single device is currently the only way to implement the DiffServ Assured Forwarding per-hop behavior (AF PHB) using Cisco IOS Software.

WRED Building Blocks



The figure shows how WRED is implemented, and the parameters that are used by WRED to influence packet-drop decisions.

The router constantly updates the WRED algorithm with the calculated average queue length, which is based on the recent history of queue lengths.

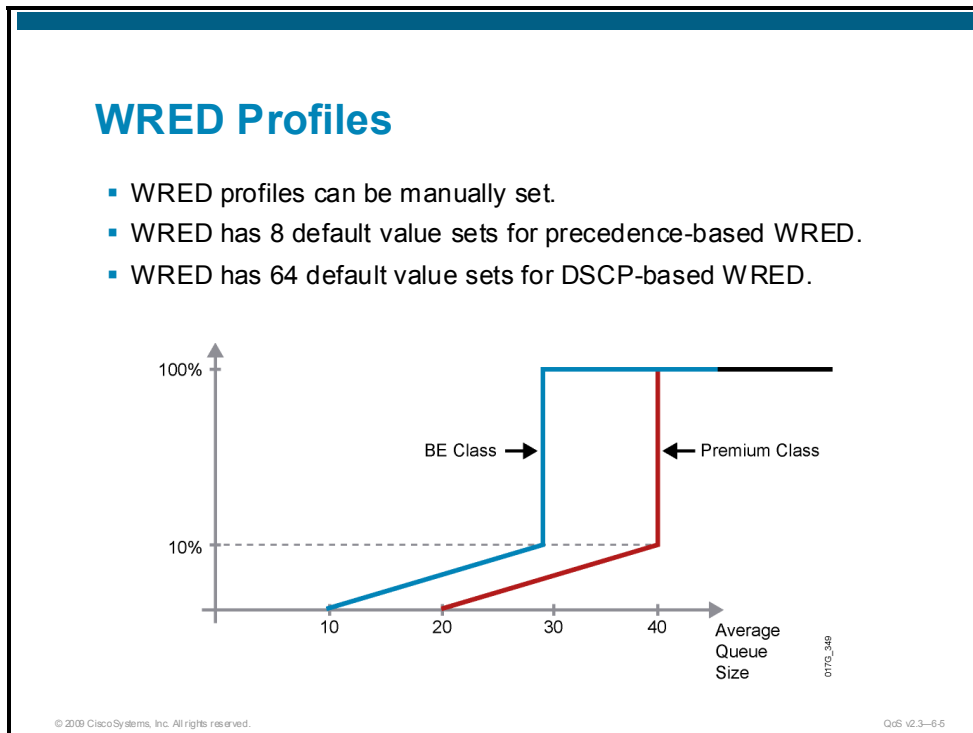
Configured in the traffic profile are the parameters that define the drop characteristics used by WRED (minimum threshold, maximum threshold, and mark probability denominator). These parameters define the WRED probability slopes.

When a packet arrives at the output queue, the IP precedence or DSCP value is used to select the correct WRED profile for the packet. The packet is then passed to WRED for processing. Based on the selected traffic profile and the average queue length, WRED calculates the probability for dropping the current packet and either drops the packet or passes it to the output queue.

If the queue is already full, the packet is tail-dropped. Otherwise, the packet will eventually be transmitted out onto the interface. If the average queue length is greater than the minimum threshold but less than the maximum threshold, based on the drop probability, WRED will either queue the packet or perform a random drop.

WRED Profiles

This topic describes the different traffic profiles that are used in WRED implementations.



This figure shows two different WRED profiles that are used for traffic of two different QoS classes (“BE” class and “Premium” class).

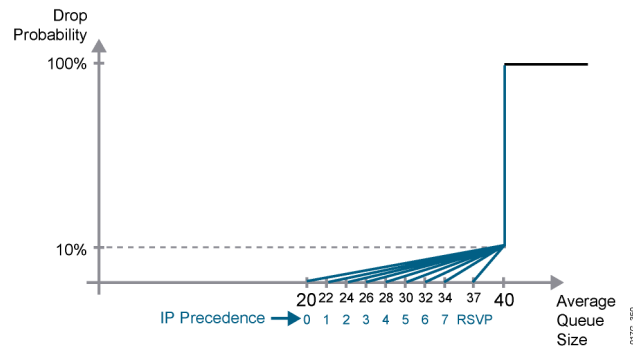
The BE traffic class has a much lower minimum (10) and maximum threshold (30). As a result, traffic belonging to the BE class will be dropped much earlier and more aggressively than traffic from the Premium class. When heavy congestion occurs, traffic belonging to the BE class will ultimately be tail dropped.

The Premium traffic class has been configured with higher minimum (20) and maximum thresholds (40). Therefore packet drop as a result of congestion will occur later (longer average queue size) and is less likely, compared to the BE class. The differences in these traffic profiles, as defined in the figure, maintain differentiated levels of service in the event of congestion.

To avoid the need for setting all WRED parameters in a router, 8 default values are already defined for precedence-based WRED, and 64 DiffServ aligned values are defined for DSCP-based WRED. Therefore, the default settings should suffice in the vast majority of deployments.

By default, the maximum threshold for all DSCP values is 40. The default mark probability denominator for all DSCP values is 10.

IP Precedence and Class Selector Profiles



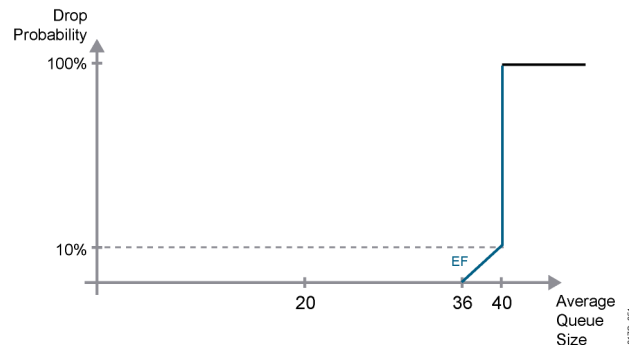
A PHB is the externally observable forwarding behavior applied at a DiffServ-compliant node to a DiffServ behavior aggregate (BA). With the ability of the system to mark packets according to DSCP setting, collections of packets (each with the same DSCP setting and sent in a particular direction) can be grouped into a DiffServ BA. Packets from multiple sources or applications can belong to the same DiffServ BA.

The class selector BA is used for backward compatibility with non-DiffServ-compliant devices (RFC 1812-compliant devices and, optionally, RFC 791-compliant devices). Therefore, the class selector range of DSCP values is used for backward compatibility with IP precedence. The same WRED profiles are applied to equal IP precedence and class selector values:

IP Precedence and Class Selector Profiles

IP Precedence	DSCP (Class Selector)	Default Minimum Threshold
0 (000)	Default (0)	20
1 (001)	CS1 (8) (001000)	22
2 (010)	CS2 (16) (010000)	24
3 (011)	CS3 (24) (011000)	26
4 (100)	CS4 (32) (100000)	28
5 (101)	CS5 (40) (101000)	30
6 (110)	CS6 (48) (110000)	32
7 (111)	CS7 (56) (111000)	34
RSVP	RSVP	37

DSCP-Based WRED (Expedited Forwarding)



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.7

In DSCP, the Expedited Forwarding (EF) PHB is identified based on these parameters:

- Ensures a minimum departure rate to provide the lowest possible delay to delay-sensitive applications
- Guarantees bandwidth to prevent starvation of the application if there are multiple applications using EF PHB
- Polices bandwidth to prevent starvation of other applications or classes that are not using this PHB

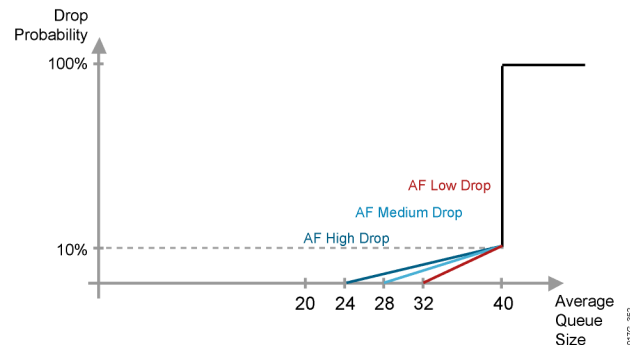
Packets requiring EF should be marked with DSCP binary value 101110 (46 or 0x2E).

For the EF DiffServ traffic class, Cisco IOS DSCP-based WRED configures itself by default so that the minimum threshold is very high, increasing the probability of no drops being applied to that traffic class. It is expected that EF traffic will be dropped very late, compared to other traffic classes, and the EF traffic is therefore prioritized in the event of congestion.

Expedited Forwarding Profile

DSCP (Six Bits)	Default Minimum Threshold
EF (101110)	36

DSCP-Based WRED (Assured Forwarding)



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.8

In DSCP, the Assured Forwarding (AF) PHB is identified based on these parameters:

- Guarantees a certain amount of bandwidth to an AF class.
- Allows access to extra bandwidth, if available.
- Packets requiring AF PHB should be marked with DSCP value “aaadd0” where “aaa” is the number of the class and “dd” is the drop preference of the traffic class.

There are four defined AF classes. Each class should be treated independently and have bandwidth allocated that is based on the QoS policy. For each AF DiffServ traffic class, Cisco IOS DSCP-based WRED configures itself by default for three different profiles, depending on the drop preference bits. All AF classes are initially marked with Drop Preference 1 (lowest drop preference), but in transit they may be marked down by policers to Drop Preference 2 or Drop Preference 3, depending if they are exceeding or violating administratively defined traffic rates.

Assured Forwarding Profiles

Assured Forwarding Class	Drop Probability	(AF Class) DSCP	Default Minimum Threshold
AF Class 1	Low Drop Prob	(AF11) 001010	32
	Medium Drop Prob	(AF12) 001100	28
	High Drop Prob	(AF13) 001110	24
AF Class 2	Low Drop Prob	(AF21) 010010	32
	Medium Drop Prob	(AF22) 010100	28
	High Drop Prob	(AF23) 010110	24
AF Class 3	Low Drop Prob	(AF31) 011010	32
	Medium Drop Prob	(AF32) 011100	28
	High Drop Prob	(AF33) 011110	24
AF Class 4	Low Drop Prob	(AF41) 100010	32
	Medium Drop Prob	(AF42) 100100	28
	High Drop Prob	(AF43) 100110	24

Configuring CBWRED

This topic describes the Cisco IOS commands that are required to configure CBWRED.

Configuring CBWRED

```
router(config)#
```

```
class-map [match-any | match-all] class-name
```

1. **Create class map**: Used for matching packets to a specified class.

```
router(config)#
```

```
policy-map policy-name
```

2. **Create policy map (service policy)**: Specifies a traffic policy that can be attached to one or more interfaces.

```
router(config-if)#
```

```
service-policy {input | output} policy-map-name
```

3. **Attach service policy**: Associates the policy map to an output interface or VC.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6.9

To configure CBWRED (WRED at the class level with CBWFQ), configure the DSCP-based and precedent-based arguments within MQC. Specific CBWRED configuration arguments are applied within a policy map. You can then apply the policy map configuration wherever policy maps are attached (for example, at the interface level, the per-virtual circuit [VC] level, or the shaper level).

The syntax shown in the figure for the **class-map**, **policy-map**, and **service-policy** commands is compatible with the syntax for those commands in Cisco IOS Software Release 12.4; however, command options may vary by router platform. For the exact command syntax for your router platform and software version, refer to the appropriate command reference on Cisco.com.

Note The WRED commands shown in this lesson apply to both IPv4 and IPv6 with no changes.

Configuring CBWRED (Cont.)

- Enables IP precedence-based WRED in the selected class within the service policy configuration mode.
- Default service profile is used.
- Command can be used at the interface, per-VC (with random-detect-group), or the class level (service-policy).
- Precedence-based WRED is the default mode.
- WRED treats non-IP traffic as precedence 0.

```
router(config-pmap-c)#
```

```
random-detect
```

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6-10

Use the **random-detect** command to enable WRED on an interface. By default, WRED is precedence-based and uses eight default WRED profiles, one for each value of IP precedence.

Within the CBWFQ system, WRED is used to perform per-queue dropping within the class queues. Therefore, each class queue has its own WRED method, which can be further weighed based on the IP precedence or DSCP value. Each queue can therefore be configured with a separate dropping policy to implement different drop policies for every class of traffic.

WRED will treat all non-IP traffic as precedence 0. As a result, non-IP traffic is more likely to be dropped than IP traffic.

If the **random-detect** command is used on versatile interface processor (VIP)-based interfaces, distributed WRED (DWRED) is enabled and the VIP CPU is responsible for WRED dropping. This can significantly increase router performance when used in the context of distributed Cisco Express Forwarding switching, which is a prerequisite for DWRED functionality. Also, DWRED can be combined with distributed weighted fair queuing (DWFWQ), enabling truly distributed queuing and congestion avoidance techniques, running independently from the central CPU.

WRED cannot be configured on the same interface as custom queuing (CQ), priority queuing (PQ), or WFQ. However, both DWRED and DWFWQ can be configured on the same interface. In addition, CBWRED can be configured in conjunction with CBWFQ. Restricting nondistributed, non-class-based WRED to only FIFO queuing on an interface is typically not a major issue because WRED is usually applied in the network core, where advanced queuing mechanisms are not typically used. WRED is suited for the network core because WRED has a relatively low performance impact on routers. Furthermore, DWRED or CBWRED can be used to overcome this limitation by combining WRED with WFQ.

The syntax for the **random-detect** command in Cisco IOS Software Release 12.4 is as follows:

random-detect [dscp-based | prec-based]

Parameter	Description
dscp-based	(Optional) Specifies that WRED is to use the differentiated services code point (DSCP) value when it calculates the drop probability for a packet. If you choose not to use either the dscp-based or the prec-based keyword, WRED uses the IP Precedence value (the default method) to calculate the drop probability for the packet.
prec-based	(Optional) Specifies that WRED is to use the IP Precedence value when it calculates the drop probability for a packet. If you choose not to use either the dscp-based or the prec-based keyword, WRED uses the IP Precedence value (the default method) to calculate the drop probability for the packet.

Changing the WRED Traffic Profile

- Changes WRED profile for specified IP precedence value.
- Packet drop probability at maximum threshold:
 - $1 / \text{mark-prob-denominator}$
- Non-weighted RED is achieved by using the same WRED profile for all precedence values.

```
router(config-pmap-c) #
```

```
random-detect precedence precedence min-threshold max-  
threshold mark-prob-denominator
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-11

When WRED is enabled, default values are selected for each traffic profile based on the weight used (IP precedence or DSCP). You can then modify these default values to match their specific administrative QoS policy goals. When modifying the default WRED profile for IP precedence, these values are configurable:

- **Minimum threshold:** When the average queue depth is above the minimum threshold, WRED starts dropping packets. The rate of packet drop increases linearly as the average queue size increases, until the average queue size reaches the maximum threshold. The default maximum threshold is equal to the default hold queue size (40) on an interface. The size of the hold queue is equivalent to the number of packets that can be held within a queue. The hold queue length ranges from 0 to 4096, and, therefore, the minimum/maximum threshold range is 1 to 4096. The default maximum threshold will reflect the defined hold queue size. Thus, if the hold queue is changed, the maximum threshold will change.
- **Maximum threshold:** When the average queue size is above the maximum threshold, all packets are dropped. If the difference between the maximum threshold and the minimum threshold is too small, many packets might be dropped at once, resulting in global synchronization.
- **Mark probability denominator:** This is the fraction of packets dropped when the average queue depth is at the maximum threshold. For example, if the denominator is 10, one out of every 10 packets is dropped when the average queue is at the maximum threshold. The maximum probability of drop at the maximum threshold can be expressed as $1 / \text{mark-prob-denominator}$. The maximum drop probability is 10 percent if default settings are used that have a mark probability denominator value of 10. The value of the mark probability can range from 1 to 65536.

If required, you can configure RED as a special case of WRED, by assigning the same profile to all eight IP precedence values. The default WRED parameter is based on the best available data. Cisco recommends that these parameters not be changed from their default values unless you have determined that your applications will benefit from the changed values.

The syntax for the **random-detect precedence** command in Cisco IOS Software Release 12.4 is as follows:

```
random-detect precedence {precedence | rsvp} min-threshold
max-threshold mark-probability-denominator
```

Parameter	Description
<i>precedence</i>	IP Precedence number. The value range is from 0 to 7. For Cisco 7000 series routers with an RSP7000 interface processor and Cisco 7500 series routers with a VIP2-40 interface processor (VIP2-50 interface processor strongly recommended), the precedence value range is from 0 to 7 only.
rsvp	Indicates Resource Reservation Protocol (RSVP) traffic.
<i>min-threshold</i>	Minimum threshold in number of packets. The value range of this argument is from 1 to 4096. When the average queue length reaches the minimum threshold, WRED randomly drops some packets with the specified IP Precedence. The default <i>min-threshold</i> depends on the precedence. The <i>min-threshold</i> for IP Precedence 0 corresponds to half of the <i>max-threshold</i> . The values for the remaining precedences fall between half the <i>max-threshold</i> and the <i>max-threshold</i> at evenly spaced intervals. For a list of the default minimum threshold values for each IP Precedence, refer to the Cisco IOS Quality of Service Solutions Command Reference on Cisco.com.
<i>max-threshold</i>	Maximum threshold in number of packets. The value range of this argument is from the value of the <i>min-threshold</i> argument to 4096. When the average queue length exceeds the maximum threshold, WRED drops all packets with the specified IP Precedence. By default, the <i>max-threshold</i> is based on the output buffering capacity and the transmission speed for the interface.
<i>mark-probability-denominator</i>	Denominator for the fraction of packets dropped when the average queue depth is at the minimum threshold. For example, if the denominator is 512, 1 out of every 512 packets is dropped when the average queue is at the minimum threshold. The value range is from 1 to 65536. The default is 10; 1 out of every 10 packets is dropped at the minimum threshold.

Changing WRED Sensitivity to Bursts

```
router(config-pmap-c) #
```

```
random-detect exponential-weighting-constant n
```

- WRED takes the average queue size to determine the current WRED mode (no drop, random drop, full drop).

$$Q_{avg}(t+1) = Q_{avg}(t) \cdot (1 - 2^{-n}) + Q_t \cdot 2^{-n}$$

- High values of N allow short bursts.
- Low values of N make WRED more burst-sensitive.
- Default value (9) should be used in most scenarios.
- Average output queue size with N=9:

$$Q_{ave}(t+1) = Q_{ave}(t) * 0.998 + Q_t * 0.002$$

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-12

WRED does not calculate the drop probability using the current queue length, but rather uses the average queue length. The average queue length is constantly recalculated using two terms: the previously calculated average queue size and the current queue size. An exponential weighting constant N influences the calculation by weighing the two terms, therefore influencing how the average queue size follows the current queue size, in this way:

- For high values of N, the previous average becomes more important. A large factor will smooth out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding drastic swings in size. The WRED process will be slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average will accommodate temporary bursts in traffic. If the value of N gets too high, WRED will not react to congestion. Packets will be transmitted or dropped as if WRED were not in effect.
- For low values of N, the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. When the queue falls below the minimum threshold, the process will stop dropping packets. If the value of N gets too low, WRED will overreact to temporary traffic bursts and drop traffic unnecessarily.

The default value of N is 9. This value should suffice for most scenarios, except perhaps those involving extremely high-speed interfaces (such as OC12), where the value can be increased slightly (to about 12) to allow more bursts.

Note

The syntax shown in the figure for the **random-detect exponential-weighting-constant** command is compatible with the syntax for the **random-detect exponential-weighting-constant** command in Cisco IOS Software Release 12.4. In that release, the default value of N is 9, and valid values range from 1 to 16. It is recommended that you use the default value of N unless you have determined that your applications would benefit from a different value.

Example: CBWFQ Using IP Precedence with CBWRED

Example: CBWFQ Using IP Precedence with CBWRED

- Enable CBWFQ to prioritize traffic according to these requirements:
 - Class **mission-critical** is marked with IP precedence values 3 and 4 (3 is high-drop, 4 is low-drop) and should get 30% of interface bandwidth.
 - Class **bulk** is marked with IP precedence values 1 and 2 (1 is high-drop, 2 is low-drop) and should get 20% of interface bandwidth.
 - All other traffic should be per-flow fair-queued.
- Use differentiated WRED to prevent congestion in all three classes.

© 2009 Cisco Systems, Inc. All rights reserved.

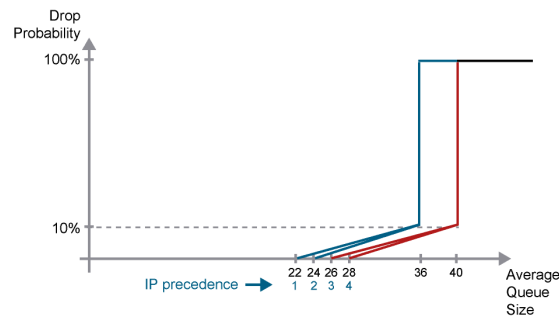
QoS v2.3-6-13

This example of CBWFQ with WRED focuses on a network that provides these three different service levels for three traffic classes:

- **Mission-critical class:** Marked with IP Precedence values 3 and 4 (3 is used for high-drop service, and 4 is used for low-drop service within the service class), and should get 30 percent of an interface bandwidth
- **Bulk class:** Marked with IP Precedence values 1 and 2 (1 is used for high-drop service, and 2 is used for low-drop service), and should get 20 percent of the interface bandwidth
- **Best-effort class:** Should get the remaining bandwidth share, and should be fair-queued

To enforce this service policy, a router will use CBWFQ to perform bandwidth sharing and WRED within service classes to perform differentiated drop.

Example: CBWFQ Using IP Precedence with CBWRED



```
class-map Mission-critical
match ip precedence 3 4
!
class-map Bulk
match ip precedence 1 2
!
policy-map Policy1
class Mission-critical
bandwidth percent 30
random-detect
random-detect precedence 3 26 40 10
random-detect precedence 4 28 40 10
```

```
class Bulk
bandwidth percent 20
random-detect
random-detect precedence 1 22 36 10
random-detect precedence 2 24 36 10
class class-default
fair-queue
random-detect
!
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-14

The figure shows the WRED traffic profile representing the QoS service policy, and the configuration that is used to implement the example service policy. The traffic is classified based on the precedence bits, and all noncontract traffic is classified into the default class, as follows:

- The mission-critical class is guaranteed at least 30 percent of bandwidth with a custom WRED profile that establishes a low-drop and a high-drop per-hop behavior.
- The bulk class is guaranteed at least 20 percent of bandwidth, is configured with somewhat lower WRED drop thresholds, and is therefore more likely to be dropped than the mission-critical class in the event of interface congestion.
- All other traffic is part of the default class and is fair-queued with default WRED parameters.

Configuring DSCP-Based CBWRED

This topic describes the Cisco IOS commands that are required to configure DSCP-based CBWRED.

Configuring DSCP-Based CBWRED

```
router(config-pmap-c)#
```

```
random-detect dscp-based
```

- Enables DSCP-based WRED.
- Command can be used at the interface, per-VC (with random-detect-group), or class level (service-policy).
- Default service profile is used.
- The WRED **random-detect** command and the WFQ **queue-limit** command are mutually exclusive for class policy.

© 2009 Cisco Systems, Inc. All rights reserved.GoS 42.3-6-15

Use the **random-detect dscp-based** command to enable DSCP-based WRED on an interface. By default, WRED is precedence-based and uses eight default WRED profiles, one for each value of IP precedence. Changing WRED weighting to values based on DSCP increases the number of WRED traffic profiles to 64.

You can configure WRED as part of the policy for a standard class or the default class. The WRED **random-detect** command and the WFQ **queue-limit** command are mutually exclusive for class policy. If you configure WRED, its packet-drop capability is used to manage the queue when packets exceeding the configured maximum count are enqueued. If you configure the WFQ **queue-limit** command for class policy, tail drop is used.

WRED cannot be configured on the same interface as CQ, PQ, or WFQ. However, both DWRED and DWFQ can be configured on the same interface. In addition, CBWRED can be configured in conjunction with CBWFQ. Restricting nondistributed, non-class-based WRED only to FIFO queuing on an interface is not a major issue because WRED is usually applied in the network core, where advanced queuing mechanisms are not typically deployed. WRED is suited for the network core because it has a relatively low performance impact on routers. Further, DWRED or CBWRED can be used to overcome this limitation by combining WRED with WFQ.

Note The **random-detect dscp-based** command (**random-detect** command with the **dscp-based** keyword) is also used to enable DSCP-based WRED in Cisco IOS Software Release 12.4.

Changing the WRED Traffic Profile

```
router(config-pmap-c) #
```

```
random-detect dscp dscpvalue min-threshold max-threshold  
mark-prob-denominator
```

- Changes WRED profile for specified DSCP value.
- Packet drop probability at maximum threshold:
 - $1 / \text{mark-prob-denominator}$

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-10

When DSCP-based WRED is enabled, default values are selected for each traffic profile based on DSCP. You can use the **random-detect dscp** command to modify these default values to match their specific administrative QoS policy goals. When modifying the default WRED profile for DSCP, these values are configurable:

- **Minimum threshold:** When the average queue depth is above the minimum threshold, WRED starts dropping packets. The rate of packet drop increases linearly as the average queue size increases, until the average queue size reaches the maximum threshold. The default maximum threshold is equal to the default hold queue size (40) on an interface. The size of the hold queue is equivalent to the number of packets that can be held within a queue. The hold queue length ranges from 0 to 4096, and, therefore, the minimum/maximum threshold range is 1 to 4096. The default maximum threshold will reflect the defined hold queue size. Thus, if the hold queue is changed, the maximum threshold will change.
- **Maximum threshold:** When the average queue size is above the maximum threshold, all packets are dropped. If the difference between the maximum threshold and the minimum threshold is too small, many packets might be dropped at once, resulting in global synchronization.
- **Mark probability denominator:** This is the fraction of packets dropped when the average queue depth is at the maximum threshold. For example, if the denominator is 10, one out of every 10 packets is dropped when the average queue is at the maximum threshold. The maximum probability of drop at the maximum threshold can be expressed as $1 / \text{mark-prob-denominator}$. The maximum drop probability is 10 percent if default settings are used that have a mark probability denominator value of 10. The value of the mark probability can range from 1 to 65536.

The syntax shown in the figure for the **random-detect dscp** command is compatible with the syntax for the **random-detect dscp** command in Cisco IOS Software Release 12.4.

Note	The default WRED parameters are based on the best available data. Cisco recommends that these parameters not be changed from their default values.

Example: CBWRED Using DSCP with CBWFQ

In this example of CBWFQ with WRED, the service policy was implemented using classes of service based on IP precedence.

Example: CBWRED Using DSCP with CBWFQ

- Enable CBWFQ to prioritize traffic according to these requirements:
 - Class **mission-critical** is marked using DSCP AF2 and should get 30% of interface bandwidth.
 - Class **bulk** is marked using DSCP AF1 and should get 20% of interface bandwidth.
 - All other traffic should be per-flow fair-queued.
- Use differentiated WRED to prevent congestion in all three classes.
- Make sure the new configurations still conform to the design and implementation from the previous example.

© 2009 Cisco Systems, Inc. All rights reserved.

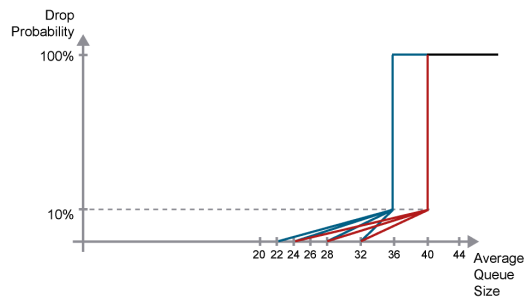
QoS v2.3-6-17

The same service policy will be configured in this example. In this case, however, DSCP-based classes of service are used. Remember that the DiffServ model itself provides defined traffic classes and their associated PHB. DiffServ-based classification is used in this example as follows:

- **Mission-critical class:** Marked using DSCP AF class 2, and should get 30 percent of an interface bandwidth.
- **Bulk class:** Marked using DSCP AF class 1, and should get 20 percent of the interface bandwidth.
- **Best-effort class:** Traffic should get the remaining bandwidth share, and should be fair-queued.

To enforce this service policy, a router will use CBWFQ to perform bandwidth sharing, and will use WRED within service classes to perform differentiated drop.

Example: CBWRED Using DSCP with CBWFQ (Cont.)



```

class-map Mission-critical
 match ip dscp af21 af22 af23 cs2
!
class-map Bulk
 match ip dscp af11 af12 af13 cs1
!
policy-map Policy1
 class Mission-critical
  bandwidth percent 30
  random-detect dscp-based
  random-detect dscp af21 32 40 10
  random-detect dscp af22 28 40 10
  random-detect dscp af23 24 40 10
  random-detect dscp cs2 24 40 10
 class Bulk
  bandwidth percent 20
  random-detect dscp-based
  random-detect dscp af11 32 36 10
  random-detect dscp af12 28 36 10
  random-detect dscp af13 24 36 10
  random-detect dscp cs1 22 36 10
 class class-default
  fair-queue
  random-detect dscp-based
!

```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3-6-18

The configuration example shows how traffic classification is performed using DSCP-based classes, representing the mission-critical class as the AF1 class, and the AF2 class as the bulk class. WRED DSCP-based parameters are set reflecting the class-dependent drop strategy, as follows:

- The mission-critical class is guaranteed at least 30 percent of bandwidth, with a custom WRED profile that establishes three different drop probabilities for AF class 2.
- The bulk class is guaranteed at least 20 percent of bandwidth, is configured with three different drop probabilities for AF class 1, and has a somewhat lower WRED maximum threshold. As a result, bulk-class traffic is more likely to be dropped than the mission-critical class in the event of interface congestion.

All other traffic is part of the default class, and is fair-queued, with default WRED parameters.

Monitoring CBWRED

This topic describes the Cisco IOS commands that are required to monitor CBWRED.

Monitoring CBWRED

router#

```
show policy-map interface interface-name
```

- Displays the packet statistics and configuration of all classes configured for all service policies on the specified interface.

```
router# show policy-map interface Ethernet 0/0
Ethernet0/0
  Service-policy output: Policy1
    Class-map: Mission-critical (match-all)
      0 packets, 0 bytes 5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 2 Match: ip dscp 18 20 22
      Weighted Fair Queueing
        Output Queue: Conversation 265
        Bandwidth 30 (%) Bandwidth 3000 (kbps)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
        exponential weight: 9
        mean queue depth: 0
      Dscp      Transmitted Random drop Tail drop   Minimum   Maximum   Mark
      (Prec)    pkts/bytes  pkts/bytes  pkts/bytes threshold threshold probability
      0 (0)      0/0         0/0         0/0         20         40        1/10
      1          0/0         0/0         0/0         22         40        1/10
      2          0/0         0/0         0/0         24         40        1/10
      ...
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.10

Use the **show policy-map interface** command to display the configuration of all classes configured for all service policies on the specified interface. This includes all WRED parameters implementing the drop policy on the specified interface.

The table explains some of the key fields of the output of the **show policy-map interface** command.

show policy-map interface Output Fields

Field	Description
service-policy output	Name of the output service policy applied to the specified interface or VC.
class-map	Class of traffic being displayed. Output is displayed for each configured class in the policy. The choice for implementing class matches (for example, match-all or match-any) can also appear next to the traffic class.
match	Match criteria specified for the class of traffic. Choices include criteria such as IP precedence, IP DSCP value, Multiprotocol Label Switching (MPLS) experimental value, access groups, and QoS groups.
exponential weight	Exponent used in the average queue size calculation for a WRED parameter group.
mean queue depth	Average queue depth based on the actual queue depth on the interface and the exponential weighting constant. It is a fluctuating average. The minimum and maximum thresholds are compared against this value to determine drop decisions.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- WRED combines RED with IP precedence or DSCP and performs packet dropping based on IP precedence or DSCP markings.
- Each WRED profile defines the minimum and maximum threshold and the maximum drop probability. Profiles are already defined by default for IP precedence and DSCP.
- The **random-detect** command is used to enable WRED on an interface.
- The **random-detect dscp-based** command is used to enable DSCP-based WRED on an interface.
- The **show policy-map interface** command displays the configuration of all classes configured for all service policies on the specified interface. This includes all WRED parameters implementing the drop policy on the specified interface.

Configuring Explicit Congestion Notification

Overview

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottleneck points. You achieve congestion avoidance through packet dropping, by using more complex techniques than simple tail drop. With the addition of explicit congestion notification (ECN) extensions to IP, routers have an alternative method of indicating congestion to peers.

This lesson describes the concept of ECN and the Cisco IOS commands that are required to configure and monitor ECN.

Objectives

Upon completing this lesson, you will be able to configure ECN to enhance the congestion avoidance features of WRED. This ability includes being able to meet these objectives:

- Describe the ECN extensions to IP
- Identify characteristics of the ECN field in IP
- Explain how ECN interacts with WRED
- Identify the Cisco IOS commands that are required to configure ECN extensions to WRED
- Identify the Cisco IOS commands that are required to monitor ECN-enabled WRED

Explicit Congestion Notification

This topic describes the ECN extensions to IP.

Explicit Congestion Notification

- TCP congestion controls are not suited to applications that are sensitive to delay or packet loss.
- ECN (RFC 3168) removes the need to rely on packet loss as a congestion indicator.
- ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value.
- Routers and end hosts can use ECN marking as a signal that the network is congested and send packets at a slower rate.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.2

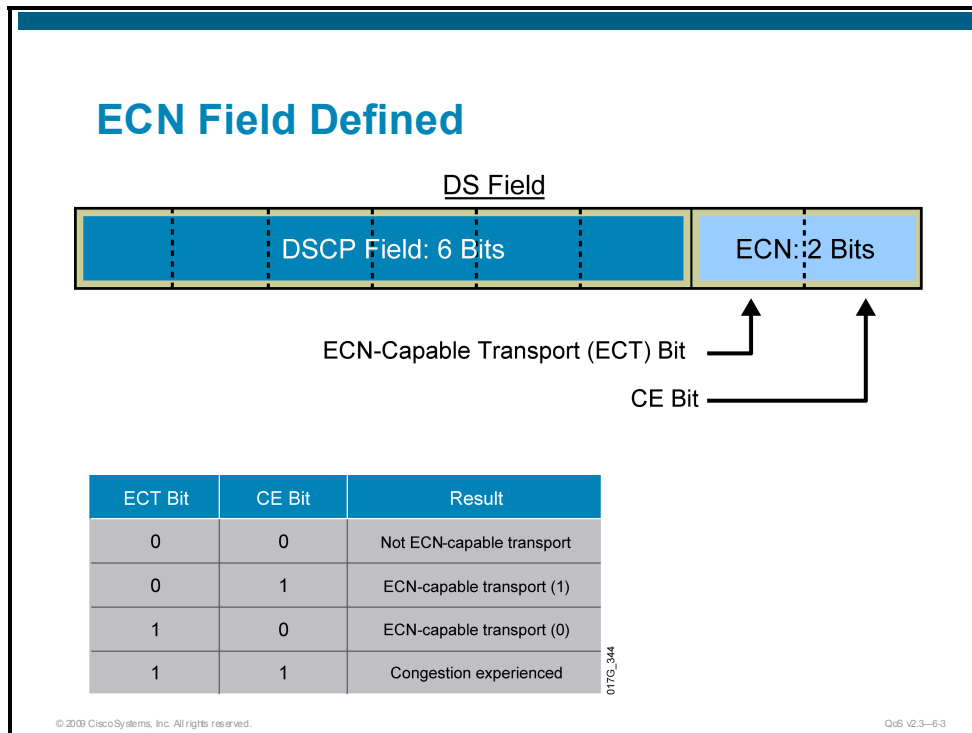
TCP determines how many unacknowledged packets it can send by gradually increasing the number of packets the session sends until it experiences a dropped packet; this is known as the TCP window size. As a result, TCP tends to cause router queues to build up at network bottleneck points. When queues become full, tail drop begins dropping all incoming packets until there is room in the queue. Tail drop does not provide differential treatment and therefore some of the fragile flow packets, sensitive to latency, may be dropped. In addition, tail drop can lead to global synchronization of packet loss across multiple flows.

Active queue management mechanisms such as random early detection (RED) or weighted random early detection (WRED) detect congestion before queues fill and overflow. Through the use of selective packet discard, these mechanisms provide congestion indication to end nodes. Therefore, active queue management (congestion avoidance) mechanisms can reduce queuing delays for all traffic sharing a specific queue. In addition, active queue management means that it is no longer necessary to rely on buffer overflow as the only means of indicating congestion.

Traditional active queue management mechanisms, such as RED, rely on the use of packet drops to indicate congestion. Packet dropping in these mechanisms is based on the average queue length exceeding a predefined threshold, rather than only when queues overflow. However, because packets are dropped prior to queues actually overflowing, the router dropping the packet is not always constrained by memory limitations and needs to actually drop the packet. With the *Addition of Explicit Congestion Notification to IP* (RFC 3168), active queue management allows routers to signal that congestion has been experienced by the router, instead of relying on the use of packet drops. Through the use of signaling congestion, aggressive flows can be slowed, thus reducing the impact of congestion and packet loss on latency-sensitive flows.

ECN Field Defined

This topic describes the characteristics of the ECN field in IP.



The Addition of Explicit Congestion Notification to IP (RFC 3168) redefines the DiffServ field (formerly the type of service [ToS] byte) to contain an ECN-specific field. The ECN field consists of the last two low-order bits of the DiffServ field, and consists of the ECN-capable transport (ECT) bit and the congestion experienced (CE) bit.

The ECT bit and the CE bit can be used to make four ECN field combinations: 00, 01, 10, and 11. The different ECT and CE bit combinations in the ECN field have these meanings:

- **00:** The ECN field combination indicates that a packet is not using ECN.
- **01 and 10:** The ECN field combinations, called ECT(1) and ECT(0), respectively, are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers will treat these two field combinations identically. Data senders can use either one or both of these two combinations.
- **11:** The ECN field combination indicates to the endpoints that congestion has been experienced. Packets arriving at a full queue of a router will be dropped.

ECN and WRED

This topic describes how ECN interacts with WRED.

ECN and WRED

- ECN is an extension to WRED.
- Congestion in WRED is indicated based on the average queue length exceeding a specific threshold value.
- If the number of packets in the queue is below the minimum threshold, packets are transmitted.
 - Treatment is identical to a network using only WRED.
- If the number of packets in the queue is above the maximum threshold, packets are tail-dropped.
 - Treatment is identical to a network using only WRED.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.4

On Cisco IOS routers, ECN is an extension to WRED functionality. WRED is an active queue management mechanism that uses packet drops as a congestion indicator to endpoints. Packets are dropped by WRED based on the average queue length exceeding a specific set of predefined threshold values (minimum and maximum threshold). ECN is an extension to WRED, in that ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value. When ECN is configured with WRED, routers and end hosts use this marking as a signal that the network is congested and slow down the rate at which packets are sent.

ECN must be interoperable with non-ECN-compliant devices. Because ECN is configured as an extension to WRED, packets are treated differently by WRED when ECN has been enabled.

If the average queue length is below the defined WRED minimum threshold, all packets are queued and transmitted normally. This behavior is identical to devices that are configured to use non-ECN-enabled WRED.

If the average queue length is greater than the maximum threshold, packets are tail-dropped. This behavior is identical to devices configured to use non-ECN-enabled WRED.

ECN and WRED (Cont.)

- If the number of packets in the queue is between the minimum and maximum threshold, one of these scenarios can occur:
 - ECN-capable endpoints and WRED determine that the packet should be dropped based on the drop probability; the ECN bits are marked, and the ECN process begins.
 - If the endpoints are not ECN-capable, the packet may be dropped based on the WRED drop probability.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.5

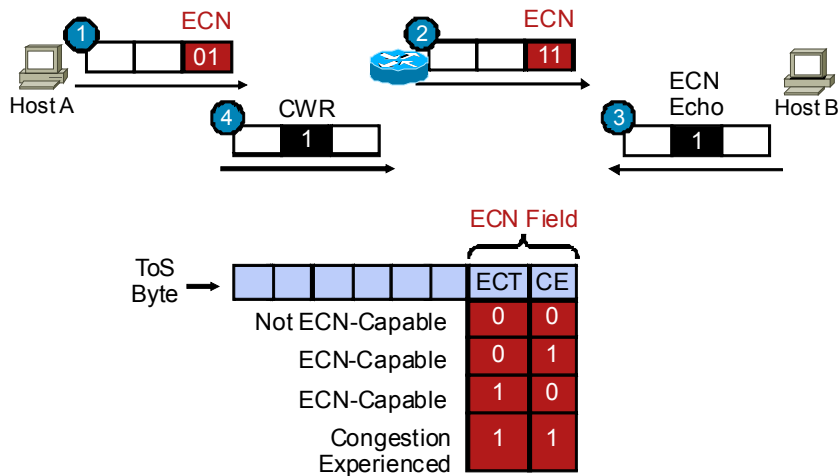
If the number of packets in the queue is below the minimum threshold, packets are transmitted. This happens whether or not ECN is enabled, and this treatment is identical to the treatment a packet receives when WRED only is being used on the network.

If the number of packets in the queue is between the minimum threshold and the maximum threshold, one of these three scenarios can occur:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1) and the WRED algorithm determines that the packet should have been dropped based on the drop probability, the ECN process is used instead of the packet being dropped.
- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet may be dropped based on the WRED drop probability. This is the identical treatment that a packet receives when WRED is enabled without ECN configured on the router.

If the number of packets in the queue is above the maximum threshold, packets are dropped. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router.

ECN and WRED Process



The following steps (illustrated in the figure) explain how ECN signaling can prevent packet drops by causing a host that is transmitting TCP packets to reduce its transmission rate:

- Step 1** The sender (Host A) sets the ECN bits in the IP headers of all data packets to 01 or 10 to indicate to the network that it is capable of participating in ECN.
- Step 2** If the following conditions exist, the router sets the ECN bits to 11 (Congestion Experienced) and sends the packet on to the receiver (Host B):
- The network is congested, or congestion is imminent.
 - The WRED algorithm determines that the packet should be dropped based on the drop probability.
 - The ECN field in the packet indicates that the endpoints are ECN-capable.
- Step 3** When Host B sees the "11" Congestion Experienced marking, it sets the ECN-Echo (ECE) flag in the TCP header of the next packet it sends back to Host A. The purpose of this flag is to tell Host A to slow down its transmissions.
- Step 4** When Host A receives the packet with the ECN Echo flag set, it reduces its congestion window to slow down its transmission rate. Host A then sets the Congestion Window Reduced (CWR) flag in the TCP header of the first new packet that it sends to Host B to inform Host B that it has reduced its window and slowed transmission.

Configuring ECN-Enabled WRED

This topic describes the Cisco IOS commands that are required to configure ECN extensions to WRED.

Configuring ECN-Enabled WRED

```
router(config-pmap-c)#  
random-detect ecn
```

- Enables ECN.
- ECN can be used whether WRED is based on the IP precedence or DSCP value.
- ECN must be configured through MQC.

```
router(config)# policy-map MyPolicy  
router(config-pmap)# class class-default  
router(config-pmap-c)# bandwidth percent 70  
router(config-pmap-c)# random-detect  
router(config-pmap-c)# random-detect ecn
```

© 2003 Cisco Systems, Inc. All rights reserved. QoS v2.3-6.7

The ECN feature must be configured through the Modular QoS command-line interface (CLI), or MQC. ECN is configured as part of a policy map after CBWRED has been enabled. ECN can be used whether the CBWRED configuration is based on IP precedence or differentiated services code point (DSCP).

The syntax shown in the figure for the **random-detect ecn** command is the same as the syntax for the **random-detect ecn** command in Cisco IOS Software Release 12.4.

Note The WRED ECN feature was introduced in Cisco IOS Software Release 12.2(8)T.

Monitoring ECN-Enabled WRED

This topic describes the Cisco IOS commands that are required to monitor ECN-enabled WRED.

Monitoring ECN-Enabled WRED

router#

```
show policy-map [policy-map]
```

- Displays the configuration of all classes for a specified service policy map or all classes for all existing policy maps

```
router# show policy-map
Policy Map MyPolicy
Class class-default
Weighted Fair Queueing
Bandwidth 70 (%)
exponential weight 9
explicit congestion notification
class min-threshold max-threshold mark-probability
-----
0          -          -          1/10
1          -          -          1/10
2          -          -          1/10
3          -          -          1/10
. . .
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.8

The **show policy-map** command displays the configuration of a service policy map created using the **show policy-map** command. The **show policy-map** command will display ECN marking information only if ECN is enabled on the interface. The table explains some of the key fields of the output of the **show policy-map** command.

show policy-map Output Fields

Field	Description
explicit congestion notification	Indication that ECN is enabled
class	IP precedence value
min-threshold	Minimum WRED threshold in number of packets
max-threshold	Maximum WRED threshold in number of packets
mark-probability	Fraction of packets dropped when the average queue depth is at the maximum threshold

Monitoring ECN-Enabled WRED (Cont.)

router#

```
show policy-map interface interface-name
```

- Displays the configuration of all classes configured for all service policies on the specified interface

```
router#show policy-map interface Serial4/1
Serial4/1
Service-policy output:policy_ecn
Class-map:prec1 (match-all)
1000 packets, 125000 bytes
30 second offered rate 14000 bps, drop rate 5000 bps
Match:ip precedence 1
Weighted Fair Queueing
Output Queue:Conversation 42
Bandwidth 20 (%)
Bandwidth 100 (kbps)
(pkts matched/bytes matched) 989/123625
(depth/total drops/no-buffer drops) 0/455/0
exponential weight:9
explicit congestion notification
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.9

The **show policy-map interface** command displays the configuration of all classes configured for all service policies on the specified interface. The counters displayed after you enter the **show policy-map interface** command are updated only if congestion is present on the interface. The **show policy-map interface** command also displays ECN marking information only if ECN is enabled on the interface. The table explains some of the key fields of the output of the **show policy-map interface** command.

show policy-map interface Output Fields

Field	Description
explicit congestion notification	Indication that ECN is enabled

Monitoring ECN-Enabled WRED (Cont.)

```

mean queue depth:0
class Transmitted Random drop Tail drop Minimum Maximum Mark
      pkts/bytes  pkts/bytes  pkts/bytes  threshold threshold probability
0      0/0        0/0        0/0        20      40      1/10
1    545/68125    0/0        0/0        22      40      1/10
2      0/0        0/0        0/0        24      40      1/10
3      0/0        0/0        0/0        26      40      1/10
4      0/0        0/0        0/0        28      40      1/10
5      0/0        0/0        0/0        30      40      1/10
6      0/0        0/0        0/0        32      40      1/10
7      0/0        0/0        0/0        34      40      1/10
rsvp    0/0        0/0        0/0        36      40      1/10

      class      ECN Mark
              pkts/bytes
0              0/0
1            43/5375
2              0/0
3              0/0
4              0/0
5              0/0
6              0/0
7              0/0
rsvp          0/0
  
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3--6-10

show policy-map interface Output Fields (Cont.)

Field	Description
mean queue depth	Average queue depth based on the actual queue depth on the interface and the exponential weighting constant. It is a moving average. The minimum and maximum thresholds are compared against this value to determine drop decisions.
class	IP precedence value.
transmitted pkts/bytes	Number of packets (also shown in bytes) transmitted.
random drop pkts/bytes	Number of packets (also shown in bytes) randomly dropped when the mean queue depth is between the minimum threshold value and the maximum threshold value for the specified IP precedence value.
tail drop pkts/bytes	Number of packets dropped when the mean queue depth is greater than the maximum threshold value for the specified IP precedence value.
minimum threshold	Minimum WRED threshold in number of packets.
maximum threshold	Maximum WRED threshold in number of packets.
mark probability	Fraction of packets dropped when the average queue depth is at the maximum threshold.
ECN mark pkts/bytes	Number of packets (also shown in bytes) marked by ECN.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value.
- ECN defines two flow control bits as extensions to the DiffServ field: the ECT bit and the CE bit.
- ECN is an extension to WRED that removes the need to rely on packet loss as a congestion indicator.
- On Cisco IOS routers, the ECN feature must be configured through MQC. Use the **random-detect ecn** command.
- The **show policy-map interface** command displays ECN marking information only if ECN is enabled on the interface.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—6-11

Module Summary

This topic summarizes the key points discussed in this module.

Module Summary

- By default, routers use tail drop as a means of congestion control when an output queue is full. Tail drop treats all traffic equally and does not differentiate between classes of service. When tail drop is in effect, packets are dropped until the congestion is eliminated and the queue is no longer full.
- Congestion avoidance techniques, such as RED, monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping.
- WRED, the Cisco implementation of RED, combines the capabilities of the RED algorithm with IP precedence or DSCP.
- ECN is an extension to WRED that enables flow control and congestion signaling without requiring packet drops.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.1

Congestion management is an area of concern for all networks that require a differentiated treatment of packet flows. Active queue management mechanisms address the limitations of relying solely on TCP congestion management techniques, which simply wait for queues to overflow and then drop packets to signal that congestion has occurred.

Congestion avoidance mechanisms such as RED and WRED allow for specific packet flows to be selectively penalized and slowed by applying a traffic profile. Traffic flows are matched against this profile and transmitted or dropped, depending upon the average length of the interface output queue. In addition, RED and WRED are extremely effective tools at preventing global synchronization of many TCP traffic flows.

Another active queue management technique is ECN. ECN is an extension to WRED that allows for signaling to be sent to ECN-enabled endpoints, instructing them to reduce their packet transmission rates. ECN also provides the benefit of not requiring packet drops when the WRED drop probability indicates otherwise.

References

For additional information, refer to these resources:

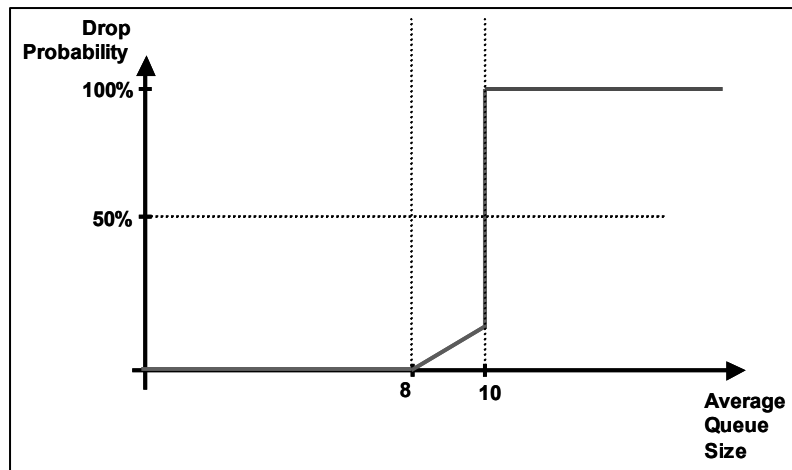
- Further details on TCP slow start can be found in RFC 2001 at this URL:
<http://www.faqs.org/rfcs/rfc2001.html>
- For a detailed discussion of TCP protocol behavior, see Geoff Huston, Telstra, “TCP Performance,” *Internet Protocol Journal*, Vol. 3, No. 2, June 2000, at this URL:
http://www.cisco.com/warp/public/759/ipj_3-2/ipj_3-2_tcp.html
- For a detailed discussion of TCP congestion behavior, see Geoff Huston, Telstra, “Future for TCP,” *Internet Protocol Journal*, Vol. 3, No. 3, September 2000, at this URL:
http://www.cisco.com/web/about/ac123/ac147/ac174/ac195/about_cisco_ipj_archive_article09186a00800c83f8.html
- For more information on Random Early Detection, refer to *Congestion Avoidance Overview* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/congestion_avoidance_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- For more information on WRED and configuring WRED, refer to *Configuring Weighted Random Early Detection* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/config_wred_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- For more information on DSCP-based WRED, refer to *DiffServ Compliant Weighted Random Early Detection* at this URL:
http://www.cisco.com/en/US/docs/ios/12_1t/12_1t5/feature/guide/dtdswred.html
- For information regarding WRED on Cisco GSR 12000 routers, refer to *Understand and Configure MDRR/WRED on the Cisco 12000 Series Internet Router* at this URL:
http://www.cisco.com/en/US/products/hw/routers/ps167/products_tech_note09186a0080094c00.shtml
- For more information on ECN, refer to *RFC 3168: The Addition of Explicit Congestion Notification to IP* at this URL: <http://www.ietf.org/rfc/rfc3168.txt>
- For more information on using WRED with ECN, refer to *WRED—Explicit Congestion Notification (ECN)* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/wred_ecn_ps6350_TSD_Products_Configuration_Guide_Chapter.html

Module Self-Check

Use the questions here to review what you learned in this module. The correct answers and solutions are found in the Module Self-Check Answer Key.

- Q1) What are two ways in which TCP manages congestion? (Choose two.) (Source: Introducing Congestion Avoidance)
- A) TCP uses tail drop on queues that have reached their queue limit.
 - B) TCP uses dropped packets as an indication that congestion has occurred.
 - C) TCP uses variable window sizes to reduce and increase the rates at which packets are sent.
 - D) TCP measures the average size of device queues and drops packets, linearly increasing the amount of dropped packets with the size of the queue.
- Q2) Two stations (A and B) are communicating using TCP. Station A has negotiated a TCP window size of 5 and as a result sends 5 packets to station B.
- Station A receives 3 ACK messages from station B indicating ACK 3.
- Which two of the following descriptions best describe the status of the communication between A and B? (Choose two.) (Source: Introducing Congestion Avoidance)
- A) Station B is acknowledging receipt of packets 1, 2, and 3, but has lost packets 4 and 5.
 - B) Station A initiates a fast-retransmit and immediately sends packet 3 to B.
 - C) Station B has not received packet 3.
 - D) Station B has received packets 1, 2, and 3, but not packet 4. It cannot be determined where packet 5 was received at B until packet 4 has been sent.
 - E) Station A will send packets 4 and 5 to station B upon receipt of the station B ACK.
- Q3) What are three important limitations of using a tail-drop mechanism to manage queue congestion? (Choose three.) (Source: Introducing Congestion Avoidance)
- A) Tail drop can cause many flows to synchronize, lowering overall link utilization.
 - B) Tail drop can cause starvation of fragile flows.
 - C) Tail drop increases the amount of packet buffer memory required, because queues must be full before congestion management becomes active.
 - D) Tail drop results in variable delays, which can interfere with delay-sensitive traffic flows.
- Q4) What are three advantages of active congestion management using RED? (Choose three.) (Source: Introducing RED)
- A) RED uses selective packet discard to eliminate global synchronization of TCP flows.
 - B) RED avoids congestion by ensuring that interface queues never become full.
 - C) RED increases the overall utilization of links.
 - D) RED uses selective packet discard to penalize aggressive flows.

- Q5) A specific RED profile has been configured with a mark probability denominator of 1. What is the effect of this configuration on packet loss as the average queue length reaches the maximum threshold? (Source: Introducing RED)
- A) Given this configuration, no packets will be dropped until the average queue length is greater than the maximum threshold.
 - B) For every active traffic flow, one packet will be discarded.
 - C) When the average queue length is at the maximum threshold, all packets are dropped.
 - D) This is an invalid configuration.
- Q6) Refer to the following RED traffic profile. In which two ways will the RED traffic profile in this figure affect the traffic flows to which it is applied? (Choose two.) (Source: Introducing RED)



RED Traffic Profile

- A) Packets may be dropped unnecessarily because the minimum threshold is too low.
 - B) This profile can result in global synchronization because the difference between the minimum and maximum thresholds is too small.
 - C) RED will not be effective because the mark probability denominator is 50 when the average queue length reaches the maximum threshold.
 - D) The reduced size of the maximum threshold will prevent tail drop and maximize link utilization.
- Q7) What are the three traffic drop modes in random early detection? (Choose three.) (Source: Introducing RED)
- A) no drop
 - B) full drop
 - C) random drop
 - D) deferred drop
- Q8) Which two QoS markers can you base the “weight” in WRED on when configuring CB-WRED? (Choose two.) (Source: Configuring Class-Based Weighted RED)
- A) CoS
 - B) DSCP
 - C) QoS group
 - D) IP precedence

- Q9) What are two requirements for configuring CB-WRED? (Choose two.) (Source: Configuring Class-Based Weighted RED)
- A) An MQC configuration that includes a policy map must be configured.
 - B) Cisco Express Forwarding must be enabled for IP.
 - C) Random detect must be enabled for DSCP-based CB-WRED.
 - D) A previous configuration of CBWFQ must be present.
- Q10) Given the following CB-WRED configuration, which command should be entered in the Bronze traffic class to properly enable CB-WRED using a minimum threshold of 22, a maximum threshold of 36, and a drop probability of 10 percent? (Source: Configuring Class-Based Weighted RED)

```
class-map Bronze
  match ip dscp cs1
!
policy-map Policy1
  class Bronze
    bandwidth percent 15
    random-detect dscp-based
  [ <- _____ -> ]
!
class class-default
  fair-queue
  random-detect dscp-based
```

- A) **random-detect dscp-based 22 36 10**
 - B) **random-detect dscp-based cs1 22 36 10**
 - C) **random-detect dscp cs1 22 36**
 - D) **random-detect dscp cs1 10 22 36**
- Q11) What are two active congestion management mechanisms available on Cisco IOS routers? (Choose two.) (Source: Configuring Class-Based Weighted RED, Configuring Explicit Congestion Notification)
- A) tail drop
 - B) weighted round robin
 - C) explicit congestion notification
 - D) weighted random early detection
- Q12) What will a router do with a newly arriving packet if its output queue is full and ECN fields are both set to a 1? (Source: Configuring Explicit Congestion Notification)
- A) drop the last packet on the queue and enqueue the newly arriving packet
 - B) perform a tail drop and drop the new packet
 - C) move the packet to the head of the queue to ensure that the receiver is signaled about the network congestion condition
 - D) allocate additional interface buffers to store the packet because it contains congestion notification information

Q13) Given the following configuration, which Cisco IOS configuration command must be added to the default class to enable ECN? (Source: Configuring Explicit Congestion Notification)

```
policy-map MyPolicy
  class class-default
    bandwidth percent 70
    random-detect
  [ <- _____ -> ]
```

- A) **wred ecn**
- B) **ecn enable**
- C) **random-detect ecn**
- D) **random-detect ecn enable**

Module Self-Check Answer Key

- Q1) B, C
- Q2) B, C
- Q3) A, B, D
- Q4) A, C, D
- Q5) C
- Q6) A, B
- Q7) A, B, C
- Q8) B, D
- Q9) A, D
- Q10) C
- Q11) C, D
- Q12) B
- Q13) C

Traffic Policing and Shaping

Overview

Within a network, different forms of connectivity can have significantly different costs to an organization. Because WAN bandwidth is relatively expensive, many organizations would like to limit the amount of traffic that specific applications can send. This is especially true when enterprise networks use Internet connections for remote site and extranet connectivity. Downloading non-business-critical images, music, and movie files can greatly reduce the amount of bandwidth available to other mission-critical applications. Traffic policing and traffic shaping are two quality of service (QoS) techniques that can be used to limit the amount of bandwidth that a specific application can use on a link.

This module describes the operation of traffic policing and traffic shaping and how these techniques can be used to rate-limit traffic. Frame Relay WANs have specific requirements; therefore, class-based traffic shaping on Frame Relay networks is also covered in this module.

Module Objectives

Upon completing this module, you will be able to use Cisco QoS traffic policing and traffic-shaping mechanisms to effectively limit the rate of network traffic. This ability includes being able to meet these objectives:

- Explain how traffic policing and traffic shaping can be used to condition traffic
- Configure class-based policing to rate-limit traffic
- Configure class-based shaping to rate-limit traffic
- Configure class-based shaping on Frame Relay WAN interfaces to rate-limit traffic

Understanding Traffic Policing and Shaping

Overview

You can use traffic policing to control the maximum rate of traffic sent or received on an interface. Traffic policing is often configured on interfaces at the edge of a network to limit traffic into or out of the network. You can use traffic shaping to control the traffic going out an interface in order to match its flow to the speed of the remote target interface and to ensure that the traffic conforms to policies contracted for it. Traffic policing and traffic shaping differ in the way they respond to traffic violations. Policing typically drops traffic, while shaping typically queues excess traffic. This lesson describes the traffic-policing and traffic-shaping quality of service (QoS) mechanisms that are used to limit the available bandwidth to traffic classes. Because both traffic policing and traffic shaping use the token bucket metering mechanism, this lesson also describes how a token bucket works.

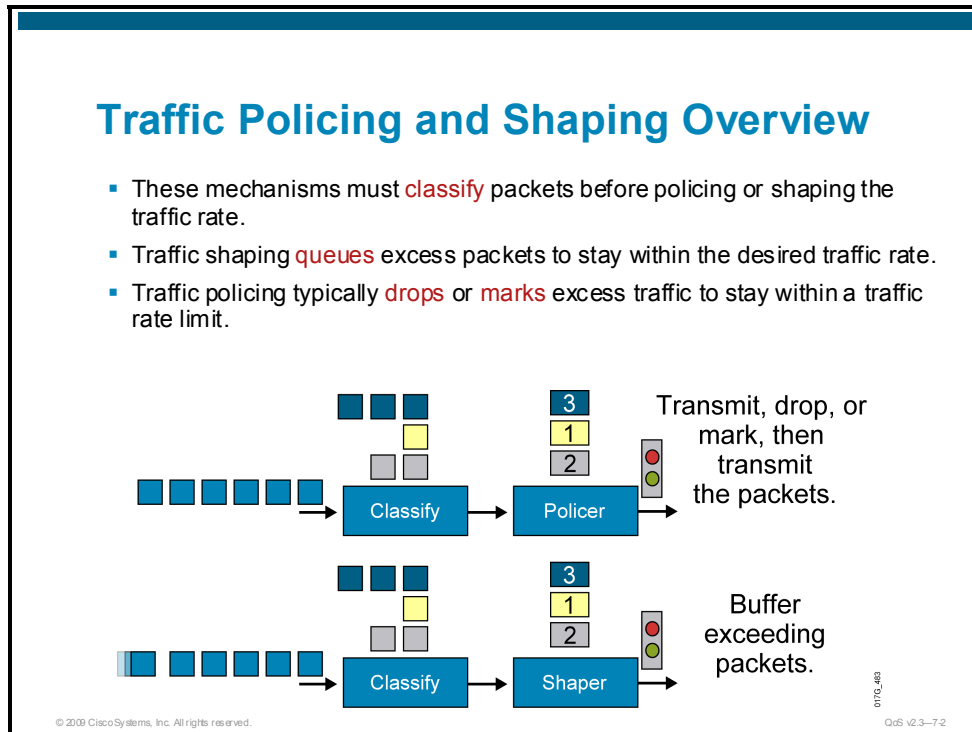
Objectives

Upon completing this lesson, you will be able to explain how to use traffic policing and traffic shaping to condition traffic. This ability includes being able to meet these objectives:

- Describe the purpose of traffic conditioning using traffic policing and traffic shaping
- List key benefits of traffic conditioning using traffic policing and traffic shaping
- Differentiate between the features of traffic policing and traffic shaping
- Explain how a token bucket can be used by network devices to measure traffic rates
- Explain how traffic can be policed using a single token bucket scheme
- Explain how traffic can be policed using a dual token bucket scheme
- Explain how traffic can be policed using a dual-rate metering scheme
- Explain how traffic can be shaped using a single token bucket scheme
- Describe the key traffic policing and shaping mechanisms available in Cisco IOS Software and how each compares to the others
- Identify the points in a network where rate-limiting can most effectively be employed

Traffic Policing and Shaping Overview

This topic describes the purpose of traffic conditioning using traffic policing and traffic shaping.



Both traffic shaping and policing mechanisms are traffic-conditioning mechanisms that are used in a network to control the traffic rate. Both mechanisms use classification so that they can differentiate traffic. They both measure the rate of traffic and compare that rate to the configured traffic-shaping or traffic-policing policy.

The difference between traffic shaping and policing can be described in terms of their implementation:

- Traffic shaping buffers excessive traffic so that the traffic stays within the desired rate. With traffic shaping, traffic bursts are smoothed out by queuing the excess traffic to produce a steadier flow of data. Reducing traffic bursts helps reduce congestion in the network.
- Traffic policing drops excess traffic in order to control traffic flow within specified rate limits. Traffic policing does not introduce any delay to traffic that conforms to traffic policies. Traffic policing can cause more TCP retransmissions, because traffic in excess of specified limits is dropped.

Traffic-policing mechanisms such as class-based policing or committed access rate (CAR) also have marking capabilities in addition to rate-limiting capabilities. Instead of dropping the excess traffic, traffic policing can alternatively mark and then send the excess traffic. This allows the excess traffic to be re-marked with a lower priority before the excess traffic is sent out. Traffic shapers, on the other hand, do not remark traffic; these only delay excess traffic bursts to conform to a specified rate.

Why Use Traffic Conditioners?

This topic describes the key benefits of traffic conditioning using traffic policing and traffic shaping.

Why Use Policing?

- To limit access to resources when high-speed access is used but not desired (sub-rate access)
- To limit the traffic rate of certain applications or traffic classes
- To mark down (re-color) exceeding traffic at Layer 2 or Layer 3

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.3

Traffic policing is typically used to satisfy one of these requirements:

- Limiting the access rate on an interface when high-speed physical infrastructure is used in transport. Rate limiting is typically used by service providers to offer customers sub-rate access. For example, a customer may have an OC-3 connection to the service provider but pay only for a T1 access rate. The service provider can rate-limit the customer traffic to T1 speed.
- Engineering bandwidth so that traffic rates of certain applications or classes of traffic follow a specified traffic rate policy. For example, rate-limiting traffic from file-sharing applications to 64 kb/s maximum.
- Re-marking excess traffic with a lower priority at Layer 2 and Layer 3, or both, before sending the excess traffic out. Cisco class-based traffic policing can be configured to mark packets at both Layer 2 and Layer 3. For example, excess traffic can be re-marked to a lower differentiated services code point (DSCP) value and also have the Frame Relay discard eligible (DE) bit set before the packet is sent out.

Why Use Shaping?

- To prevent and manage congestion in ATM and Frame Relay networks, where asymmetric bandwidths are used along the traffic path
- To regulate the sending traffic rate to match the subscribed (committed) rate in Frame Relay or ATM networks

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.4

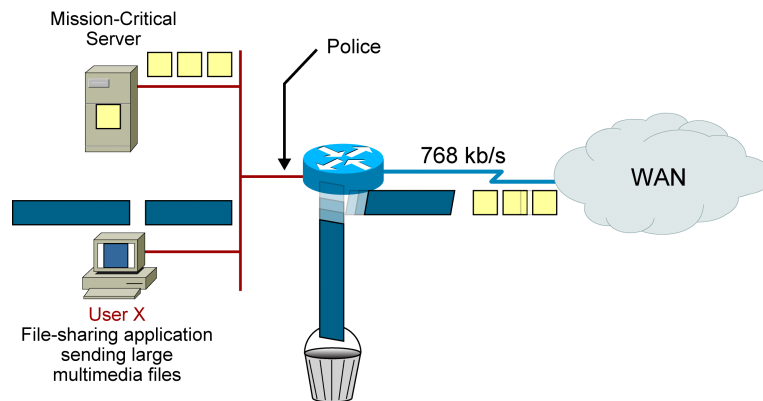
Traffic shaping is typically used for the following:

- To prevent and manage congestion in ATM and Frame Relay networks, where asymmetric bandwidths are used along the traffic path. If shaping is not used, buffering can occur at the slow (usually the remote) end, which can lead to queuing, causing delays, and overflow, causing drops.
- To prevent dropping of noncompliant traffic by the ATM or Frame Relay service provider by not allowing the traffic to burst above the subscribed (committed) rate. This allows the customer to keep local control of traffic regulation.

Example: Traffic Policing

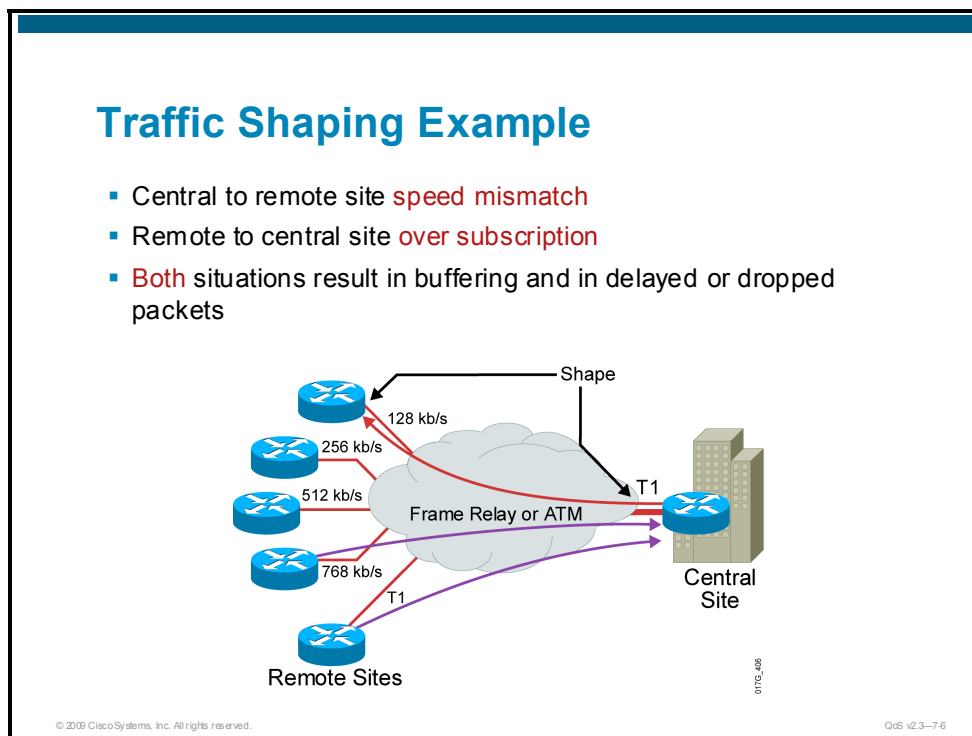
Traffic Policing Example

- Rate-limit file-sharing application traffic to 56 kb/s.
- Do not rate-limit traffic from mission-critical server.



You can use traffic policing to divide the shared resource (the upstream WAN link) between many flows. In this example, the router Fast Ethernet interface has an input traffic-policing policy applied to it, in which the mission-critical server traffic rate is not rate-limited, but the User X file-sharing application traffic is rate-limited to 56 kb/s. All file-sharing application traffic from User X that exceeds the rate limit of 56 kb/s will be dropped.

Example: Traffic Shaping



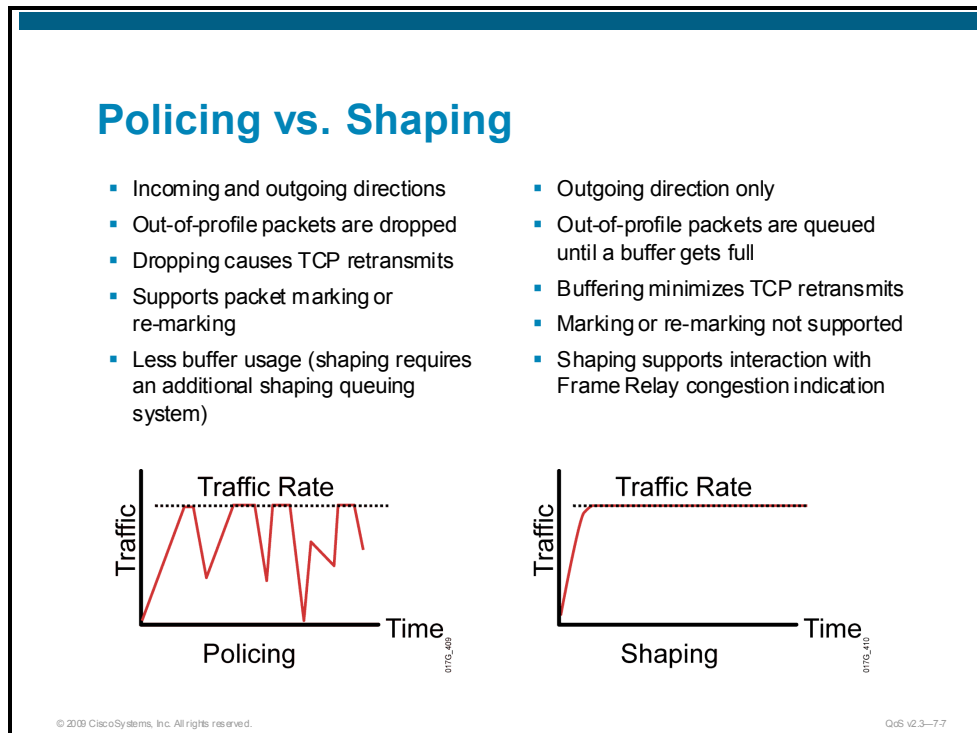
Traffic-shaping tools limit the transmit rate from a source by queuing the excess traffic. This limit is typically a value lower than the line rate of the transmitting interface. Traffic shaping can be used to account for speed mismatches that are common in nonbroadcast multiaccess (NBMA) networks such as Frame Relay and ATM.

In the figure, these two types of speed mismatches are shown:

- The central site can have a higher-speed link than the remote site. You can deploy traffic shaping at the central-site router to shape the traffic rate out of the central-site router to match the link speed of the remote site. For example, the central router can shape the permanent virtual circuit (PVC) outgoing traffic rate (going to the top remote-site router) to 128 kb/s to match that remote-site link speed. At each remote-site router, traffic shaping is also implemented to shape the remote-site outgoing traffic rate to 128 kb/s to match the committed information rate (CIR).
- The aggregate link speed of all the remote sites can be higher than the central-site link speed, thereby over-subscribing the central-site link speed. In this case, you can configure the remote-site routers for traffic shaping to avoid oversubscription at the central site. For example, you can configure the bottom two remote-site routers to shape the PVC outgoing traffic rate to 256 kb/s to avoid the central-site router from being over-subscribed.

Policing vs. Shaping

This topic describes the difference between the features of traffic policing and traffic shaping.



Shaping queues excess traffic by holding packets inside a shaping queue. Use traffic shaping to shape the outbound traffic flow when the outbound traffic rate is higher than a configured shape rate. Traffic shaping smoothes traffic by storing traffic above the configured rate in a shaping queue. Therefore, shaping increases buffer utilization on a router and causes unpredictable packet delays. Traffic shaping can also interact with a Frame Relay network, adapting to indications of Layer 2 congestion in the WAN. For example, if the backward explicit congestion notification (BECN) bit is received, the router can lower the rate limit to help reduce congestion in the Frame Relay network.

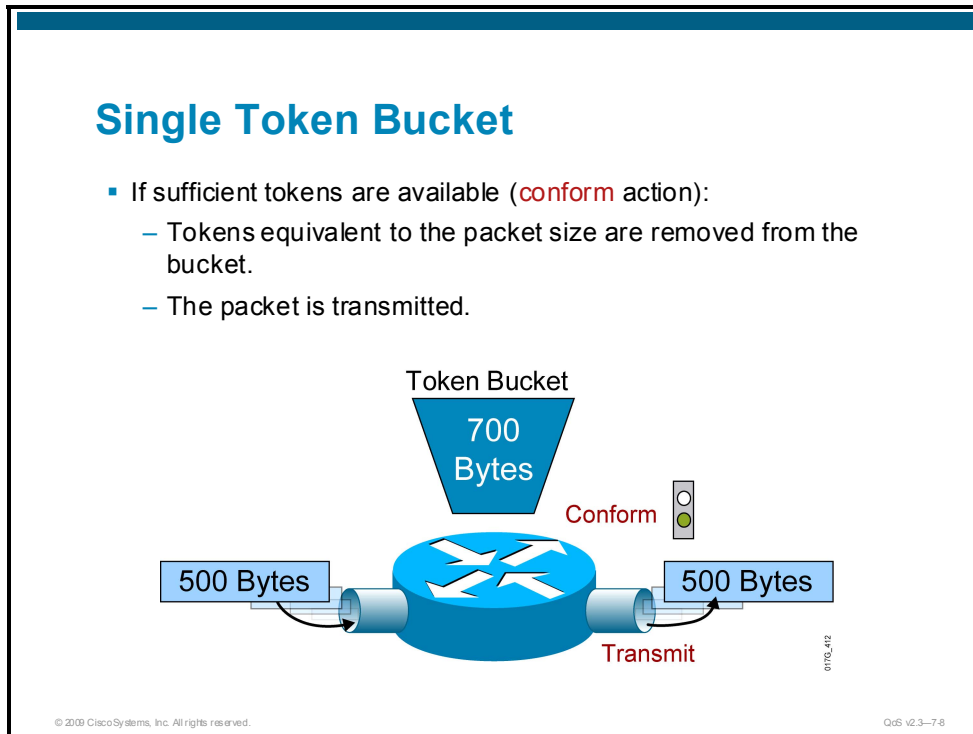
You can apply policing to either the inbound or outbound direction, while you can apply shaping only in the outbound direction. Policing drops nonconforming traffic instead of queuing the traffic like shaping. Policing also supports marking of traffic. Traffic policing is more efficient in terms of memory utilization than traffic shaping because no additional queuing of packets is needed.

Both traffic policing and traffic shaping ensure that traffic does not exceed a bandwidth limit, but each mechanism has different impacts on the traffic:

- Policing drops packets more often, generally causing more retransmissions of connection-oriented protocols such as TCP.
- Shaping adds variable delay to traffic, possibly causing jitter.

Measuring Traffic Rates

This topic describes how a token bucket can be used by network devices to measure traffic rates.



The token bucket is a mathematical model that is used by routers and switches to regulate traffic flow. The model has two basic components:

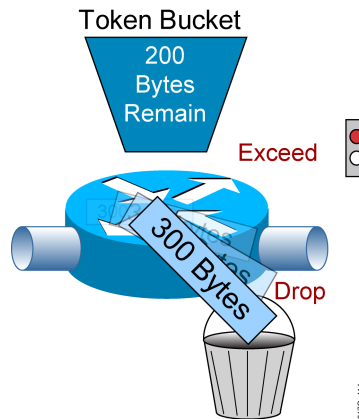
- **Tokens:** Each token represents permission to send a fixed number of bits into the network. Tokens are put into a token bucket at a certain rate by the Cisco IOS Software.
- **Token bucket:** A token bucket has the capacity to hold a specified number of tokens. Each incoming packet, if forwarded, takes tokens from the bucket, representing the packet size. If the bucket fills to capacity, newly arriving tokens are discarded. Discarded tokens are not available to future packets. If there are not enough tokens in the token bucket to send the packet, the traffic conditioning mechanisms may take these actions:
 - Wait for enough tokens to accumulate in the bucket (traffic shaping)
 - Discard the packet (traffic policing)

Using a single token bucket model, the measured traffic rate can conform or exceed the specified traffic rate. The measured traffic rate is conforming if there are enough tokens in the single token bucket to transmit the traffic. The measured traffic rate is exceeding if there are not enough tokens in the single token bucket to transmit the traffic.

The figure shows a single token bucket traffic-policing implementation. Starting with a current capacity of 700 bytes worth of tokens accumulated in the token bucket, when a 500-byte packet arrives at the interface, its size is compared to the token bucket capacity (in bytes). The 500-byte packet conforms to the rate limit ($500 \text{ bytes} < 700 \text{ bytes}$), and the packet is forwarded; 500 bytes worth of tokens are taken out of the token bucket, leaving 200 bytes worth of tokens for the next packet.

Single Token Bucket (Cont.)

- If sufficient tokens are NOT available (**exceed** action):
 - Drop (or mark) the packet



When the next 300-byte packet arrives immediately after the first packet, and no new tokens have been added to the bucket (which is done periodically), the packet exceeds the rate limit. The current packet size (300 bytes) is greater than the current capacity of the token bucket (200 bytes), and the exceed action is performed. The exceed action can be to drop or mark the packet when traffic policing.

Example: Token Bucket as a Coin Bank

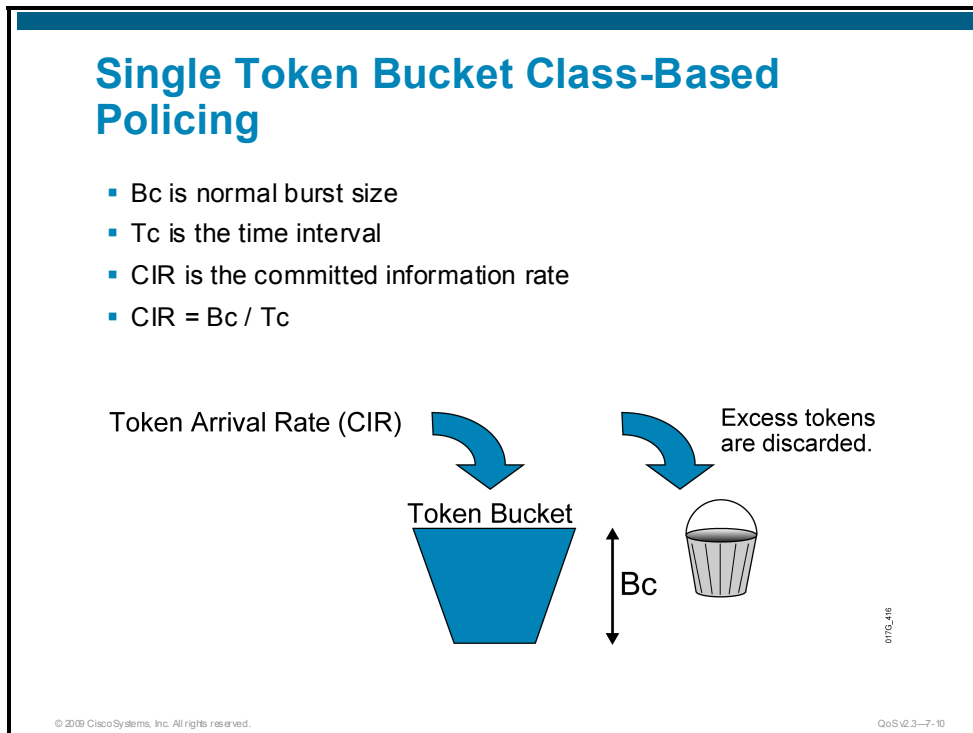
Think of a token bucket as a coin bank. Every day you can insert a coin into the bank (the token bucket). At any given time, you can only spend what you have saved up in the bank. On the average, if your saving rate is a dollar per day, your long-term average spending rate will be one dollar per day if you constantly spend what you saved. However, if you do not spend any money on a given day, you can build up your savings in the bank to the maximum that the bank can hold. For example, if the size of the bank is limited to five dollars, and if you save and do not spend for five straight days, the bank will contain five dollars. When the bank fills to its capacity, you will not be able to put any more money in it. Then, at any time, you can spend up to five dollars (bursting above the long-term average rate of one dollar per day).

Conforming rate (using the coin bank example) means that if you have two dollars in the bank and you try to spend one dollar, that is considered conforming, because you are not spending more than you have saved.

Exceeding rate (using the coin bank example) means that if you have two dollars in the bank and you try to spend three dollars, it is considered exceeding, because you are spending more than you have saved.

Single Token Bucket Class-Based Policing

This topic describes how traffic can be policed using a single token bucket scheme.



Token bucket operations rely on parameters such as CIR, committed burst (Bc), and committed time window (Tc). Bc is known as the normal burst size. The mathematical relationship between CIR, Bc, and Tc is as follows:

$$CIR \text{ (bps)} = Bc \text{ (bits)} / Tc \text{ (sec)}$$

With traffic policing, new tokens are added into the token bucket based on the interpacket arrival rate and the CIR. Every time a packet is policed, new tokens are added back into the token bucket. The number of tokens added back into the token bucket is calculated as follows:

$$(Current \text{ Packet Arrival Time} - Previous \text{ Packet Arrival Time}) * CIR$$

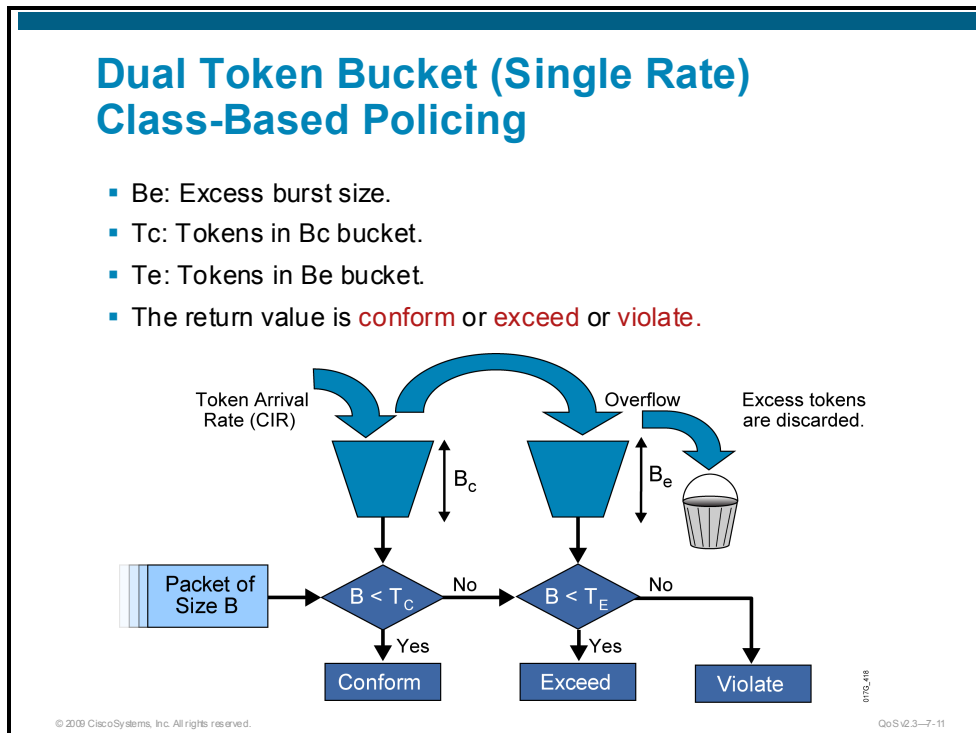
An amount (Bc) of tokens is forwarded without constraint in every time interval (Tc). For example, if 8000 bits (Bc) worth of tokens are placed in the bucket every 250 milliseconds (Tc), the router can steadily transmit 8000 bits every 250 milliseconds if traffic constantly arrives at the router.

$$CIR \text{ (normal burst rate)} = 8000 \text{ bits (Bc)} / 0.25 \text{ seconds (Tc)} = 32 \text{ kbps}$$

Without any excess bursting capability, if the token bucket fills to capacity (Bc of tokens), the token bucket will overflow and newly arriving tokens will be discarded. Using the example, in which the CIR is 32 kb/s (Bc = 8000 bits and Tc = 0.25 seconds), the maximum traffic rate can never exceed a hard rate limit of 32 kb/s.

Dual Token Bucket Class-Based Policing

This topic describes how traffic can be policed using a dual token bucket scheme.



You can configure class-based traffic policing to support excess bursting capability. With excess bursting, after the first token bucket is filled to B_c , extra (excess) tokens can be accumulated in a second token bucket. Excess burst (B_e) is the maximum amount of excess traffic over and above B_c that can be sent during the time interval *after* a period of inactivity. With a single rate-metering mechanism, the second token bucket with a maximum size of B_e fills at the same rate (CIR) as the first token bucket. If the second token bucket fills up to capacity, no more tokens can be accumulated and the excess tokens are discarded.

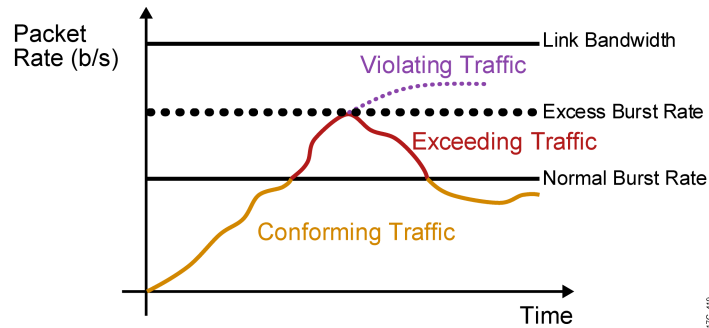
When using a dual token bucket model, the measured traffic rate can be as follows:

- **Conforming:** There are enough tokens in the first token bucket with a maximum size of B_c .
- **Exceeding:** There are not enough tokens in the first token bucket, but there are enough tokens in the second token bucket with a maximum size of B_e .
- **Violating:** There are not enough tokens in the first or second token bucket.

With dual token bucket traffic policing, the typical actions performed are sending all conforming traffic, re-marking (to a lower priority), sending all exceeding traffic, and dropping all violating traffic. The main benefit of using a dual token bucket method is the ability to distinguish between traffic that exceeds the B_c but not the B_e . This enables a different policy to be applied to packets in the B_e category. Referring to the coin bank example, think of the CIR as the savings rate (\$1 per day). B_c is how much you can save into the bank per day (\$1). T_c is the interval at which you put money into the coin bank (one day). B_e (\$5) allows you to burst over the average spending rate of one dollar per day if you are not spending a dollar per day.

Dual Token Bucket (Single Rate) Class-Based Policing (Cont.)

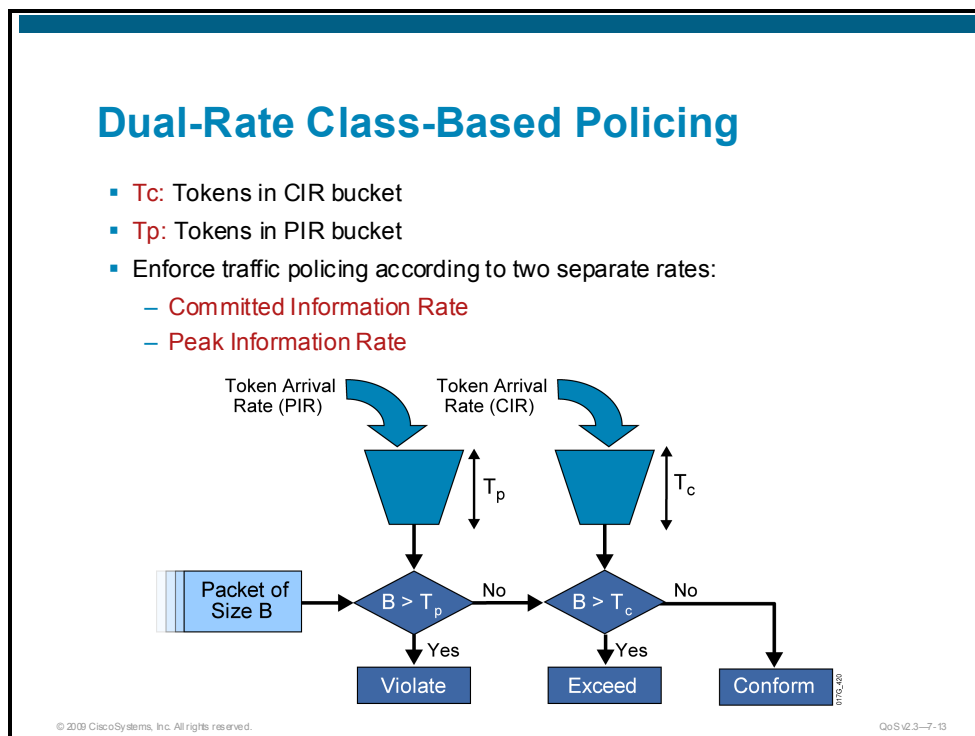
- Traffic is **conforming**, **exceeding**, or **violating**



Using a dual token bucket model allows traffic exceeding the normal burst rate (CIR) to be metered as exceeding, and traffic that exceeds the excess burst rate to be metered as violating traffic. Different actions can then be applied to the conforming, exceeding, and violating traffic.

Dual-Rate Token Bucket Class-Based Policing

This topic describes how traffic can be policed using a dual-rate metering scheme.



With dual-rate metering, traffic rate can be enforced according to two separate rates: CIR and peak information rate (PIR). Before this feature was available, you could meter traffic using a single rate based on the CIR with single or dual buckets. Dual-rate metering supports a higher level of bandwidth management and supports a sustained excess rate based on the PIR.

With dual-rate metering, the PIR token bucket fills at a rate based on the packet arrival rate, and the configured PIR and the CIR token bucket fills at a rate based on the packet arrival rate and the configured CIR.

When a packet arrives, the PIR token bucket is first checked to see if there are enough tokens in the PIR token bucket to send the packet. The violating condition occurs if there are not enough tokens in the PIR token bucket to transmit the packet. If there are enough tokens in the PIR token bucket to send the packet, then the CIR token bucket is checked. The exceeding condition occurs if there are enough tokens in the PIR token bucket to transmit the packet but not enough tokens in the CIR token bucket to transmit the packet. The conforming condition occurs if there are enough tokens in the CIR bucket to transmit the packet.

Dual-rate metering is often configured on interfaces at the edge of a network to police the rate of traffic entering or leaving the network. In the most common configurations, traffic that conforms is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs.

Dual-Rate Class-Based Policing (Cont.)

Two-rate policer marks packets as either conforming, exceeding, or violating a specified rate.

- If $(B > T_p)$, the packet is marked as violating the specified rate.
- If $(B > T_c)$, the packet is marked as exceeding the specified rate, and the T_p token bucket is updated as $T_p = T_p - B$.
- If the packet is marked as conforming to the specified rate, both token buckets (T_c and T_p) are updated as $T_p = T_p - B$ and $T_c = T_c - B$.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7-14

In addition to rate limiting, traffic policing using dual-rate metering allows marking of traffic according to whether the packet conforms, exceeds, or violates a specified rate.

The token bucket algorithm provides users with three different actions for each packet: a conform action, an exceed action, and an optional violate action. Traffic entering the interface with two-rate policing configured is placed into one of these categories. Within these three categories, users can decide packet treatments. For example, a user may configure a policing policy as follows:

- Conforming packets are transmitted. Packets that exceed may be transmitted with a decreased priority; packets that violate are dropped.
- The violating condition occurs if there are not enough tokens in the PIR bucket to transmit the packet.
- The exceeding condition occurs if there are enough tokens in the PIR bucket to transmit the packet but not enough tokens in the CIR bucket to transmit the packet. In this case, the packet can be transmitted and the PIR bucket is updated to $T_p - B$ remaining tokens, where T_p is the size of the PIR bucket and B is the size of the packet to be transmitted.
- The conforming condition occurs if there are enough tokens in the CIR bucket to transmit the packet. In this case, the packets are transmitted and both buckets (T_c and T_p) are decremented to $T_p - B$ and to $T_c - B$, respectively, where T_c is the size of the CIR bucket, T_p is the size of the PIR bucket, and B is the size of the packet to be transmitted.

Example: Dual-Rate Token Bucket as a Coin Bank

Using a dual-rate token bucket is like using two coin banks, each with a different savings rate. However, you can take out money from only one of the banks at a time.

For example, you can save ten dollars per day into the first coin bank (PIR = peak spending rate = \$10 per day) and then at the same time, you can save five dollars per day into the second bank (CIR = normal average spending rate = \$5 per day). However, the maximum amount you can spend is \$10 per day, not \$15 per day, because you can take out money from only one bank at a time.

In this example, after one day of savings, your first coin bank (PIR bucket) will contain \$10 and your second coin bank (CIR bucket) will contain \$5. The three different spending cases are examined here to show how dual-rate metering operates, using the coin bank example:

- **Case 1:** If you try to spend \$11 at once, then you are violating ($T_p < B$) your peak-spending rate of \$10 per day. In this case, you will not be allowed to spend the \$11 because \$11 is greater than the \$10 you have in the first coin bank (PIR bucket). Remember, you can only take out money from one of the banks at a time.
- **Case 2:** If you try to spend \$9 at once, then you are exceeding ($T_p > B > T_c$) your normal average spending rate of \$5 per day. In this case, you will be allowed to spend the \$9 and just the first coin bank (PIR bucket) will be decremented to $\$10 - \$9$, or \$1.

After spending \$9, the maximum amount you can continue to spend on that day is decremented to \$1.

- **Case 3:** If you try to spend \$4, then you are conforming ($T_p > B$ and $T_c > B$) to your normal average spending rate of \$5 per day. In this case, you will be allowed to spend the \$4, and both coin banks (PIR and CIR bucket) will be updated.

The first coin bank (PIR bucket) will be updated to $\$10 - \$4 = \$6$, and the second bank (CIR bucket) will be updated to $\$5 - \$4 = \$1$.

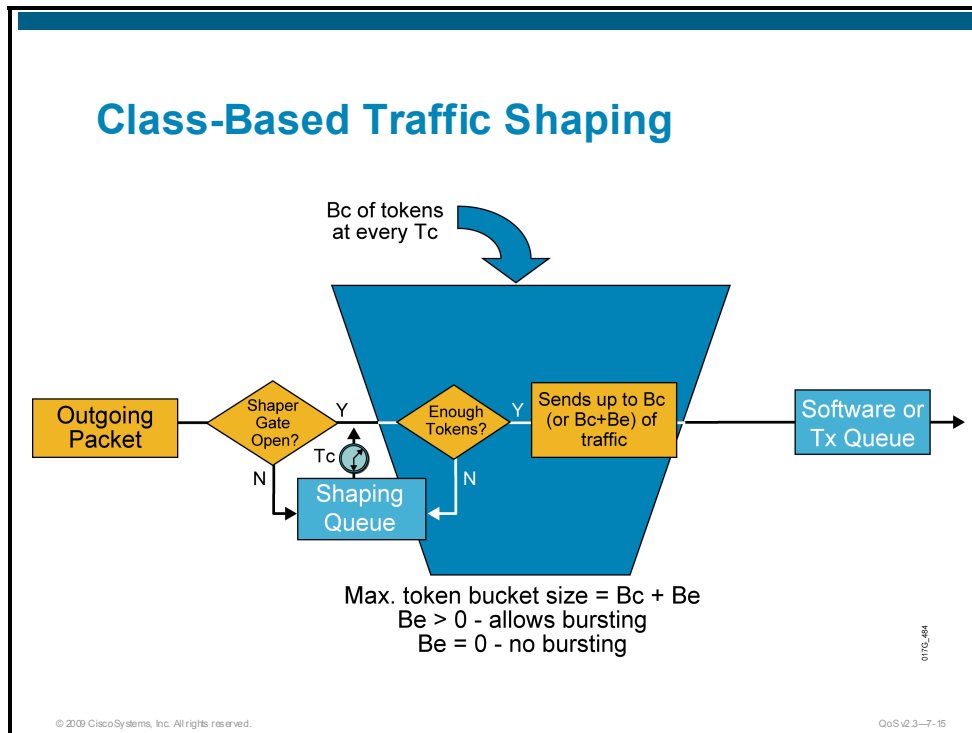
Both coin banks are updated because after spending \$4, the maximum amount you can continue to spend on that day is decremented to \$6, and the normal spending rate for that same day is decremented to \$1.

Therefore, after spending \$4, the following will occur:

- If you continue to spend \$7 on that same day, then you will be violating your peak-spending rate for that day. In this case, you will not be allowed to spend the \$7 because \$7 is greater than the \$6 you have in the first coin bank (PIR bucket).
- If you continue to spend \$5 on that same day, then you will be exceeding your normal average spending rate for that day. In this case, you will be allowed to spend the \$5 and just the first coin bank (PIR bucket) will be decremented to $\$6 - \$5$, or \$1.
- If you continue to spend 50 cents on that same day, then you will be conforming to your normal average spending rate for that day. In this case, you will be allowed to spend the 50 cents, and both coin banks (PIR and CIR bucket) will be updated. The first coin bank (PIR bucket) will be updated to $\$6 - \$0.50 = \$5.50$, and the second coin bank (CIR bucket) will be updated to $\$1 - \$0.50 = \$0.50$.

Class-Based Traffic Shaping

This topic describes how traffic can be shaped using a single token bucket scheme.



Cisco class-based traffic shaping only applies for outbound traffic.

Class-based traffic shaping uses the basic token bucket mechanism, in which Bc of tokens are added at every Tc time interval. The maximum size of the token bucket is Bc + Be. You can think of the traffic shaper operation like opening and closing of a transmit gate at every Tc interval. If the shaper gate is opened, the shaper checks to see if there are enough tokens in the token bucket to send the packet. If there are enough tokens, the packet is immediately forwarded. If there are *not* enough tokens, the packet is queued in the shaping queue until the next Tc interval. If the gate is closed, the packet is queued behind other packets in the shaping queue.

For example, on a 128-kb/s link, if the CIR is 96 kb/s, the Bc is 12 kb/s, the Be is 0, and the Tc = 0.125 seconds, then during each Tc (125 ms) interval, the traffic shaper gate opens and up to 12 KB can be sent. To send 12 KB over a 128-kb/s line will only take 91.25 ms. Therefore the router will, on the average, be sending at three-quarters of the line rate (128 kb/s * 3/4 = 96 kb/s).

Traffic shaping also includes the ability to send more than Bc of traffic in some time intervals after a period of inactivity. This extra number of bits in excess to the Bc is called Be.

Cisco IOS Traffic Policing and Shaping Mechanisms

This topic describes the key traffic policing and shaping mechanisms available in Cisco IOS Software and how each compares to the others.

Cisco IOS Traffic-Policing Mechanisms	
	Class-Based Policing
Enable Method	Enabled in policy map
Conditions Actions	Conform, exceed, violate Drop, set, transmit
Implementations	Single or dual token bucket Single or dual rate policing Multi actions

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-7-16

The figure lists the characteristics of the class-based traffic-policing mechanism that is available in Cisco IOS Software. Class-based policing is also available on some Cisco Catalyst switches.

Class-based policing supports a single or dual token bucket. Class-based policing also supports single-rate or dual-rate metering and multiaction policing. Multiaction policing allows more than one action to be applied; for example, marking the Frame Relay DE bit and also the DSCP value before sending the exceeding traffic.

Class-based policing is configured using Modular QoS command-line interface (CLI), or MQC, with the **police** command under the policy map.

Cisco IOS Traffic-Shaping Mechanisms

	CB Shaping	DTS	FRTS
Restriction	Shaper for any Subinterface	Shaper on VIP	Shaper for FR Only
Classification	Class-Based	Subinterface or Group or Class - Based	Per DLCI or Subinterface
Link Fragmentation and Interleaving	No Support for FRF.12	Supports FRF.12	Supports FRF.12
Frame Relay Support	Understands BECN and FECN	Understands BECN and FECN	Understands BECN and FECN
Configuration	Supported via MQC	Supported via MQC	Supported via MQC

0176_806

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7-17

The figure lists some of the different traffic-shaping mechanisms available in Cisco IOS Software: the newer class-based traffic shaping, Distributed Traffic Shaping (DTS), and FRTS.

Class-based traffic shaping uses MQC to allow traffic to be shaped per traffic class as defined by the class map. You can use class-based traffic shaping in combination with class-based weighted fair queuing (CBWFQ), in which the shaped rate is used to define an upper rate limit while the bandwidth statement within the CBWFQ configuration is used to define a minimum rate limit.

DTS is a feature that is specific to the higher-end platforms, such as the Cisco 7500 or the Cisco 12000 series routers. These platforms have the ability to offload traffic shaping from the main processor to the individual interface processor (VIP or line card). In networks in which distributed Cisco Express Forwarding is the preferred mode of switching, DTS on the VIP or line card is the logical choice for implementing traffic shaping.

FRTS is used to shape Frame Relay traffic only. FRTS allows an individual PVC (data-link connection identifier [DLCI]) to be shaped. FRTS can use priority queuing (PQ), custom queuing (CQ), or weighted fair queuing (WFQ) as the shaping queue and only supports FIFO as the software queue.

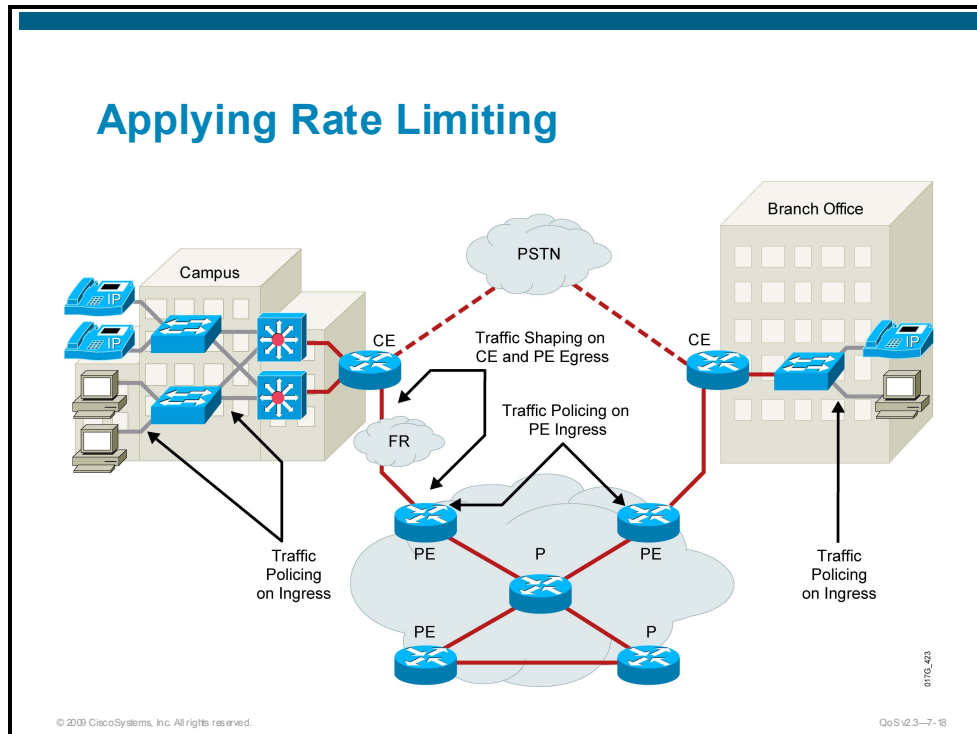
Both DTS and FRTS support FRF.12 Frame Relay fragmentation, while class-based shaping does not support FRF.12 fragmentation for Frame Relay.

All these traffic-shaping mechanisms can interact with a Frame Relay network, adapting to indications of Layer 2 congestion in the WAN. For example, if the BECN bit is received, the router can lower the rate limit to help reduce congestion in the Frame Relay network. And if the forward explicit congestion notification (FECN) bit is received, the router can generate a test frame with the BECN bit set. This enables the sender to notice congestion even if there is no data traffic flowing back from the receiver to the sender.

Only class-based shaping configurations are described in this module.

Applying Traffic Conditioners

This topic describes the points in a network where rate-limiting can most effectively be employed.

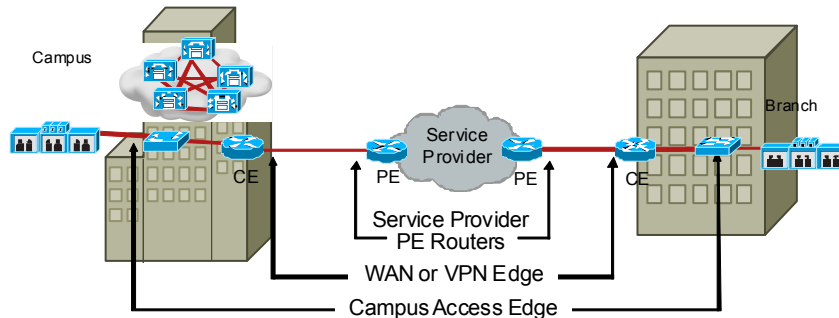


In a typical enterprise network, traffic policing is often implemented at the access or distribution layer to limit certain traffic classes before that traffic exits the campus onto the WAN. Traffic shaping is often implemented at the WAN edge when there are speed mismatches or oversubscription.

In a typical service provider network, traffic policing is often implemented inbound at the provider edge (PE) router to rate-limit incoming traffic from the customer edge (CE) router to ensure that the customer traffic rate is not exceeding the contractual rate. Traffic shaping is often implemented outbound at the PE and at the CE to limit the traffic rate between the PE and CE and to allow for FRF.12 fragmentation on Frame Relay connections between the PE and the CE.

Traffic Policing and Shaping with Cisco TelePresence

- Policing Cisco TelePresence traffic should generally be avoided whenever possible. Exceptions include the following:
 - At the WAN or VPN edge
 - At the service provider PE routers, in the ingress direction
 - At the campus access edge
- It is recommended to avoid shaping Cisco TelePresence flows unless absolutely necessary



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7-22

Although some exceptions exist, policing Cisco TelePresence traffic should generally be avoided whenever possible. Cisco TelePresence is highly sensitive to drops (with a 0.05% packet loss target); therefore, policing its traffic rates could be extremely detrimental to its flows and could ultimately ruin the high level of user experience that it is intended to deliver.

However, there are three places where Cisco TelePresence traffic may be legitimately policed over the network.

- **At the WAN or VPN edge:** The first place where Cisco TelePresence traffic may be legitimately policed automatically occurs if Cisco TelePresence is assigned to a Low-Latency Queue (LLQ) within Cisco IOS routers at the WAN or VPN edge. This is because any traffic assigned to an LLQ is automatically policed by an implicit policer set to the exact value as the LLQ rate. For example, if Cisco TelePresence is assigned an LLQ of 15 Mb/s, it is also implicitly policed by the LLQ algorithm to exactly 15 Mb/s; any excess Cisco TelePresence traffic is dropped.

Note The implicit **policer** within the LLQ feature is **active only when LLQ is active**. LLQ never engages unless the link is physically congested or a hierarchical QoS shaper forces LLQ to engage prior to physical link congestion. Therefore, the implicit policer of LLQ never engages unless there is physical congestion on the link or a hierarchical QoS shaper forces it to engage prior to physical link congestion.

- **At the service provider PE routers, in the ingress direction:** The second most common place that Cisco TelePresence is likely to be policed in the network is at the service provider's PE routers, in the ingress direction. Service providers must police traffic classes, especially real-time traffic classes, to enforce service contracts and prevent possible oversubscription on their networks and thus ensure service-level agreements.

At the campus access edge: The third (and optional) place where policing Cisco TelePresence may prove beneficial in the network is at the campus access edge. You can deploy access-edge policers for security purposes to mitigate the damage caused by the

potential abuse of trusted switch ports. Because Cisco TelePresence endpoints can mark its flows to the recommended 802.1Q/p CoS value (CoS 4) and DSCP codepoint value (CS4), you may choose to trust the CoS or DSCP values received from these ports.

However, if disgruntled employees gain physical access to the Cisco TelePresence switch ports, they may send whatever traffic they choose to over these ports, and their flows are trusted over the network. Such rogue traffic flows may hijack voice or video queues and easily ruin call or video quality over the QoS-provisioned network infrastructure.

Therefore, you may choose to limit the scope of damage that such network abuse may present by configuring access-edge policers on Cisco TelePresence switch ports to re-mark or drop out-of-profile traffic originating on these ports. Supporting this approach, RFC 4594 recommends edge policing the Real-Time Interactive service class via a single-rate policer.

Note	For more information on access edge policers, refer to the <i>Enterprise QoS Solution Reference Network Design Guide</i> on Cisco.com. The recommended burst parameter for TelePresence policers is discussed in detail in the “Branch QoS Design for TelePresence” section of the <i>Cisco TelePresence Network Systems 2.0 Design Guide</i> on Cisco.com.

It is recommended to avoid shaping Cisco TelePresence flows unless absolutely necessary because of the QoS objective of shapers themselves. Specifically, the role of shapers is to delay traffic bursts above a certain rate and to smooth out flows to fall within contracted rates. Sometimes this is done to ensure that traffic rates are within a carrier’s Committed Information Rate (CIR); other times, shaping is performed to protect other data classes from a bursty class.

Shapers temporarily buffer traffic bursts above a given rate and as such introduce jitter as well as absolute delay. Because Cisco TelePresence is so sensitive to delay and especially jitter, it is recommended not to shape Cisco TelePresence flows.

If your objective is to meet the CIRs of a carrier, you can achieve your objective by properly provisioning the adequate bandwidth and burst allowances on the circuit. If your objective is to protect other traffic classes from Cisco TelePresence bursts, explicitly protecting each class with a guaranteed minimum bandwidth rate (such as a Cisco IOS CBWFQ) is a better approach than using shaping. In either case, a shaper would be a sub-optimal tool to meet the desired objective and would cause quality issues on the Cisco TelePresence flows.

The Cisco TelePresence traffic queue (whether you choose to place it in a CBWFQ or a second strict priority LLQ) must be provisioned with the proper mean rate (bits per second) and burst allowance (burst bytes exceeding the mean).

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Traffic shaping and policing are mechanisms that use classification to limit traffic rate.
- Traffic shaping queues excess packets to stay within the contractual rate. Traffic policing typically drops excess traffic to stay within the limit; alternatively, it can re-mark, then send excess traffic.
- Both traffic policing and shaping ensure that traffic does not exceed a bandwidth limit, but they have different impacts on the traffic.
- Traffic rate is metered using a token bucket mathematical model.
- With traffic policing, new tokens are added into the token bucket based on the CIR.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—7-23

Summary (Cont.)

- Class-based traffic policing can be configured to support excess bursting capability. With excess bursting, after the first token bucket is filled to Bc, extra (excess) tokens can be accumulated in a second token bucket.
- With dual-rate metering, traffic rate can be enforced according to two separate rates: CIR and PIR.
- Class-based traffic shaping uses the basic token bucket mechanism, in which Bc of tokens are added at every Tc time interval.
- Class-based policing is the latest Cisco IOS traffic-policing mechanism. Class-based shaping, DTS, and FRTS are three Cisco IOS traffic-shaping mechanisms.
- Policing is often implemented at the access or distribution layer, while shaping is implemented at the WAN edge.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—7-24

Configuring Class-Based Policing

Overview

Cisco IOS Software supports two different traffic-policing mechanisms: committed access rate (CAR) and class-based policing. CAR is an older Cisco traffic-policing feature, and class-based policing is a newer Cisco traffic-policing mechanism based on the modular quality of service (QoS) command-line interface (CLI), or Modular QoS CLI (MQC). Cisco recommends that you use MQC features when possible to implement QoS in the network. You should avoid using traffic-policing configurations with CAR, because no new features or functionality are planned. However, Cisco will continue to support CAR for existing implementations.

This lesson describes the configuration tasks that are used to implement class-based traffic policing to rate-limit certain traffic classes.

Objectives

Upon completing this lesson, you will be able to configure class-based policing to rate-limit traffic. This ability includes being able to meet these objectives:

- Explain the key features of class-based policing
- Identify the Cisco IOS commands required to configure single-rate class-based policing
- Identify the Cisco IOS commands required to configure dual-rate class-based policing
- Identify the Cisco IOS commands required to configure percentage-based class-based policing
- Identify the Cisco IOS commands used to monitor class-based policing

Class-Based Policing Overview

This topic describes the key features of class-based policing.

Class-Based Policing Overview

- Class-based policing is used to rate-limit a traffic class to a configured bit rate.
- Class-based policing can **drop** or **re-mark** and transmit exceeding traffic.
- Class-based policing can be implemented using a **single** or **dual** token bucket scheme.
- Class-based policing supports multiaction policing:
 - Applying two or more set parameters as a conform or exceed or violate action
- Class-based policing conforms to two RFCs:
 - RFC 2697, “A Single Rate Three Color Marker”
 - RFC 2698, “A Two Rate Three Color Marker”
- Class-based policing is configured using the **MQC** method.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.2

The class-based policing feature performs these functions:

- Limits the input or output transmission rate of a class of traffic based on user-defined criteria
- Marks packets by setting different Layer 2 or Layer 3 markers, or both

You can implement class-based policing using a single or double token bucket method as the metering mechanism. When the violate action option is not specified in the **police** MQC command, the single token bucket algorithm is engaged; when the violate action option is specified in the **police** MQC command, the dual token bucket algorithm is engaged.

A dual token bucket algorithm allows traffic to do the following:

- Conform to the rate limit when the traffic is within the average bit rate
- Exceed the rate limit when the traffic exceeds the average bit rate, but does not exceed the allowed excess burst
- Violate the rate limit when the traffic exceeds both the average rate and the excess bursts

Depending on whether the current packet conforms with, exceeds, or violates the rate limit, one or more actions can be taken by class-based policing, as follows:

- **Transmit:** The packet is transmitted.
- **Drop:** The packet is dropped.

- **Set IP Precedence (or DSCP value) and transmit:** The IP precedence or differentiated services code point (DSCP) bits in the packet header are rewritten. The packet is then transmitted. This action can be used to either color (set precedence) or recolor (modify existing packet precedence) the packet.
- **Set QoS group and transmit:** The QoS group is set and the packet forwarded. Because the QoS group is only significant within the local router (that is, the QoS group is not transmitted outside the router), the QoS group setting is used in later QoS mechanisms, such as class-based weighted fair queuing (CBWFQ), and performed in the same router on an outgoing interface.
- **Set MPLS Experimental bits and transmit:** The Multiprotocol Label Switching (MPLS) Experimental bits are set. The packet is then transmitted. These are usually used to signal QoS parameters in an MPLS cloud.
- **Set Frame Relay DE bit and transmit:** The Frame Relay discard eligible (DE) bit is set in the Layer 2 (Frame Relay) header and the packet is transmitted. This setting can be used to mark excessive or violating traffic (which should be dropped with preference on Layer 2 switches) at the edge of a Frame Relay network.
- **Set ATM CLP bit and transmit:** The ATM cell loss priority (CLP) bit is set in the Layer 2 (ATM) header and the packet is transmitted. This setting can be used to mark excessive or violating traffic (which should be dropped with preference on Layer 2 switches) at the edge of an ATM network.

Multiaction policing is a mechanism that can apply more than one action to a packet; for example, setting the DSCP as well as the CLP bit on the exceeding packets.

Class-based policing also supports single- or dual-rate metering. With the two-rate policer, traffic policing can be enforced according to two separate rates: committed information rate (CIR) and peak information rate (PIR). You can specify the use of these two rates, along with their corresponding values, by using two keywords, **cir** and **pir**, in the **police** command.

Cisco class-based policing mechanisms conform to these two DiffServ RFCs:

- **RFC 2697, *A Single Rate Three Color Marker*:** The Single Rate Three Color Marker meters an IP packet stream and marks its packets to one of three states: conform, exceed, or violate. Marking is based on a CIR and two associated burst sizes, a committed burst (Bc) size and an excess burst (Be) size. A packet is marked conform if it does not exceed the Bc, exceed if it does exceed the Bc but not the Be, and violate otherwise.
- **RFC 2698, *A Two Rate Three Color Marker*:** The Two Rate Three Color Marker meters an IP packet stream and marks its packets to one of three states: conform, exceed, or violate. A packet is marked violate if it exceeds the PIR. Otherwise a packet is marked either exceed or conform, depending on whether it exceeds or does not exceed the CIR. This is useful, for example, for ingress policing of a service where a peak rate needs to be enforced separately from a committed rate.

Configuring Single-Rate Class-Based Policing

This topic describes the Cisco IOS Software commands that are required to configure single-rate class-based policing.

Configuring Single-Rate Class-Based Policing

```
router (config-pmap-c)#
```

```
police avg-rate [BC [BE]] [conform-action action] [exceed-action action] [violate-action action]
```

- **avg-rate:** Traffic rate in b/s (8000 to 200,000,000)
- **B_C:** normal burst sets the size in **bytes**
 - Default is 1500 bytes or CIR / 32, whichever is higher
- **B_E:** Excess burst sets the size in **bytes**
 - Default is BC
- **Action:**
 - transmit (default conform action)
 - drop (default exceed and violate action)
 - set-prec-transmit ip-precedence
 - set-dscp-transmit dscp
 - set-qos-transmit qos-group
 - set-mpls-exp-transmit mpls-exp
 - set-frde-transmit
 - set-clp-transmit

© 2003 Cisco Systems, Inc. All rights reserved.QoS v2.3-7.3

The MQC-based **police** command defines policing parameters for a specific traffic class. The **avg-rate** parameter defines the policed CIR; Bc and Be define the token bucket sizes in bytes; and the action defines an action for conforming, exceeding, and optionally violating traffic.

If Bc (in bytes) is not specified, it will default to CIR / 32, or 1500 bytes, whichever is higher. When using the formula CIR / 32 to calculate the default Bc (in bytes), Cisco IOS Software uses a Tc of 0.25 second, where:

$$B_c \text{ (in bytes)} = (CIR \times T_c) / 8$$

$$B_c \text{ (in bytes)} = (CIR \times 0.25 \text{ seconds}) / 8 = CIR / 32$$

If Be (in bytes) is not specified, it will default to Bc. In a single token bucket case, Cisco IOS Software ignores the Be value. This means that excess bursting is disabled.

The Be rate can be specified when a violate action is configured, therefore using a dual token bucket. This allows Be to be explicitly configured instead of using the default value of Be = Bc. Be specifies the size of the second (excess) token bucket.

Dual token bucket policing with the violate action was introduced in Cisco IOS Software Release 12.1(5)T.

The syntax for the **police** command in Cisco IOS Software Release 12.4 is as follows:

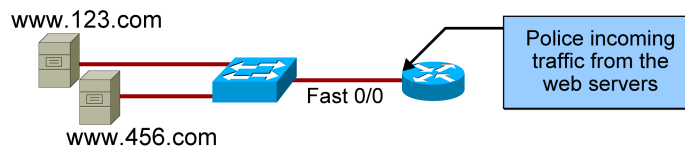
```
police bps [burst-normal] [burst-max] conform-action action  
exceed-action action [violate-action action]
```

Syntax Description

Parameter	Description
<i>bps</i>	Average rate, in bits per second. Valid values are 8000 to 200,000,000.
<i>burst-normal</i>	(Optional) Normal burst size in bytes. Valid values are 1000 to 51,200,000. Default normal burst size is 1500.
<i>burst-max</i>	(Optional) Maximum burst size, in bytes. Valid values are 1000 to 51,200,000. Default varies by platform.
conform-action	Specifies action to take on packets that conform to the rate limit.
exceed-action	Specifies action to take on packets that exceed the rate limit.
violate-action	(Optional) Specifies action to take on packets that violate the normal and maximum burst sizes.

Parameter	Description
<i>action</i>	Action to take on packets. Specify one of the following keywords: ■ drop—Drops the packet. ■ set-clp-transmit value—Sets the ATM Cell Loss Priority (CLP) bit from 0 to 1 on the ATM cell and transmits the packet with the ATM CLP bit set to 1. ■ set-cos-inner-transmit value—Sets the inner class of service field as a policing action for a bridged frame on the Enhanced FlexWAN module when using bridging features on SPAs with the Cisco 7600 SIP-200 and Cisco 7600 SIP-400 on the Cisco 7600 series router. ■ set-cos-transmit value—Sets the COS packet value and sends it. ■ set-discard-class-transmit—Sets the discard class attribute of a packet and transmits the packet with the new discard class setting. ■ set-dscp-transmit value—Sets the IP differentiated services code point (DSCP) value and transmits the packet with the new IP DSCP value. ■ set-dscp-tunnel-transmit value—Sets the DSCP value (0 to 63) in the tunnel header of a Layer 2 Tunnel Protocol Version 3 (L2TPv3) or Generic Routing Encapsulation (GRE) tunneled packet for tunnel marking and transmits the packet with the new value. ■ set-frde-transmit value—Sets the Frame Relay Discard Eligibility (DE) bit from 0 to 1 on the Frame Relay frame and transmits the packet with the DE bit set to 1. ■ set-mpls-experimental-imposition-transmit value—Sets the Multiprotocol Label Switching (MPLS) experimental (EXP) bits (0 to 7) in the imposed label headers and transmits the packet with the new MPLS EXP bit value. ■ set-mpls-experimental-topmost-transmit value—Sets the MPLS EXP field value in the topmost MPLS label header at the input and/or output interfaces. ■ set-prec-transmit value—Sets the IP precedence and transmits the packet with the new IP precedence value. ■ set-prec-tunnel-transmit value—Sets the precedence value (0 to 7) in the tunnel header of an L2TPv3 or GRE tunneled packet for tunnel marking and transmits the packet with the new value. ■ set-qos-transmit value—Sets the qos-group value and transmits the packet with the new qos-group value. ■ transmit—Transmits the packet. The packet is not altered.

Class-Based Policing Example: Single-Rate, Single Token Bucket



```
class-map www.123.com
  match source-address mac 000d.ddd.f.0480
!
class-map www.456.com
  match source-address mac 000d.dddc.ad21
!
policy-map ServerFarm
  class www.123.com
    police 512000 conform-action transmit exceed-action drop
  class www.456.com
    police 256000 conform-action transmit exceed-action drop
!
interface FastEthernet 0/0
  service-policy input ServerFarm
```

The class-based policing configuration example shows two configured traffic classes based on upstream media access control (MAC) addresses. Traffic from the particular web server, which is classified by its MAC address, is policed to a fixed bandwidth with no excess burst capability using a single token bucket. Conforming traffic is sent as-is and exceeding traffic is dropped. In this case, the `www.123.com` web server is policed to a rate of 512 kb/s, and the `www.456.com` web server is policed to a rate of 256 kb/s.

Because the violate action is not specified, this will use a single token bucket scheme and no excess bursting is allowed.

In this example, the committed burst size (Bc) is not specified, and therefore it will default to the 512,000 / 32 (16,000 bytes) and 256,000 / 32 (8000 bytes), respectively.

The default Bc setting can be examined by showing the policy map, as follows. Notice that the Be is not displayed because no excess bursting is allowed using a single token bucket with class-based policing.

```
router#show policy-map ServerFarm
Policy Map ServerFarm
Class www.123.com
  police cir 512000 bc 16000
    conform-action transmit
    exceed-action drop
Class www.456.com
  police cir 256000 bc 8000
    conform-action transmit
    exceed-action drop
```

Example: Single Rate, Dual Token Bucket Class-Based Policing

Class-Based Policing Example: Single-Rate, Dual Token Bucket

```
class-map www.123.com
match source-address mac 000d.dddf.0480
!
class-map www.456.com
match source-address mac 000d.dddc.ad21
!
policy-map ServerFarm
class www.123.com
  police 512000 conform-action set-prec-transmit 4 exceed-action
  set-prec-transmit 3 violate-action drop
class www.456.com
  police 256000 conform-action set-prec-transmit 4 exceed-action
  set-prec-transmit 3 violate-action drop
!
interface FastEthernet 0/0
service-policy input ServerFarm
```

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-7.5

The class-based policing configuration example shows two configured traffic classes based on upstream MAC addresses.

Traffic from the particular web server, which is classified by its MAC address, is policed to a fixed bandwidth with excess burst capability using a dual token bucket, by configuring a violate action. Conforming traffic will be sent as-is, exceeding traffic will be marked to IP Precedence 3 and transmitted, and all violating traffic will be dropped.

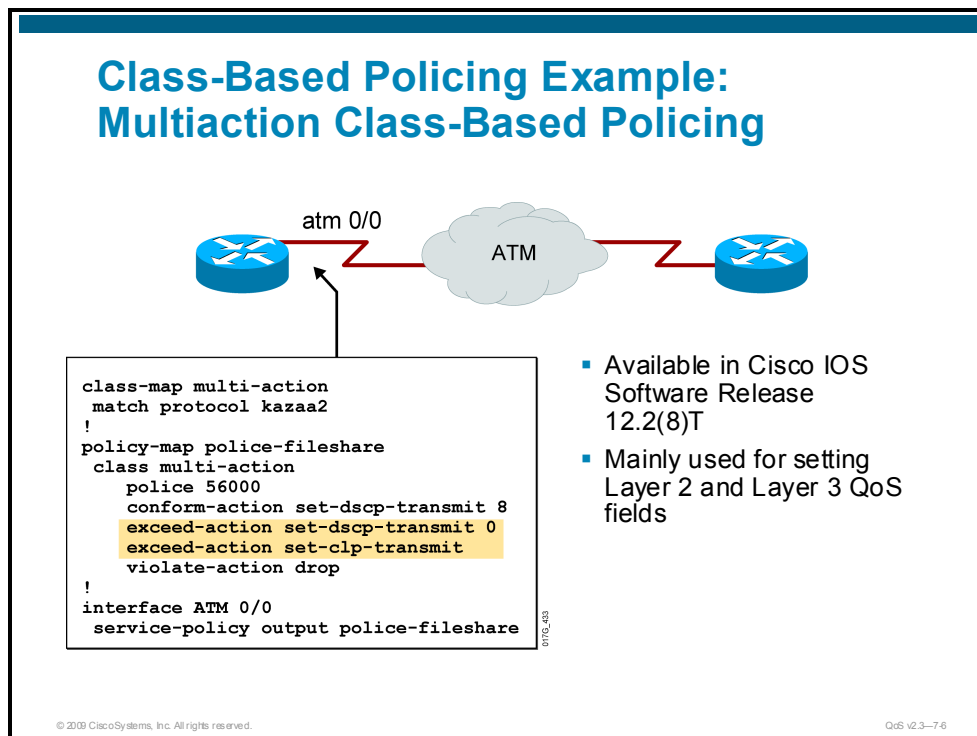
In this example, because the violate action is specified, a dual token bucket scheme with excess bursting will be used. The committed burst size (Bc) is not specified, and therefore it will default to the 512,000 / 32 (16,000 bytes) and 256,000 / 32 (8000 bytes), respectively. The excess burst size (Be) is also not specified, and therefore it will default to Bc.

The default Bc and Be settings can be examined by showing the policy map, as follows:

```
router#show policy-map ServerFarm
Policy Map ServerFarm
Class www.123.com
  police cir 512000 bc 16000 be 16000
    conform-action transmit
    exceed-action set-prec-transmit 3
    violate-action drop
Class www.456.com
  police cir 256000 bc 8000 be 8000
    conform-action transmit
```

```
exceed-action set-prec-transmit 3
violate-action drop
```

Example: Multiaction Class-Based Policing



The class-based policing configuration is an example of a multiaction class-based policing.

In this case, all Kazaa2 traffic is policed to 56 kb/s. All conforming traffic will be marked with a DSCP value of 8, and then transmitted. All exceeding traffic will be marked with a DSCP value of 0, and the CLP bit in the ATM header will also be set before it is transmitted. All violating traffic will be dropped.

The multiaction feature was introduced in Cisco IOS Software Release 12.2(8)T and is primarily used so that class-based policing can mark at Layer 2 and Layer 3 at the same time.

Configuring Dual-Rate Class-Based Policing

This topic describes the Cisco IOS Software commands that are required to configure dual-rate class-based policing.

Configuring Dual-Rate Class-Based Policing

```
router (config-pmap-c) #
```

```
police {cir cir} [bc conform-burst] {pir pir} [be peak-burst] [conform-action action] [exceed-action action] [violate-action action]
```

- Specifies both the CIR and the PIR for two-rate traffic policing
- CIR = committed information rate (b/s)
- PIR = peak information rate (b/s)
- The Bc and Be keywords and their associated arguments (conform-burst and peak-burst, respectively) are optional
- Available in Cisco IOS Software Release 12.2(4)T

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.7

Dual-rate class-based policing was introduced in Cisco IOS Software Release 12.2(4)T. With dual-rate policing, traffic policing can be enforced according to two separate rates: CIR and PIR. The use of these two rates can be specified, along with their corresponding values, by using two keywords, **cir** and **pir**, in the **police** command. The Bc and Be keywords and their associated arguments (**conform-burst** and **peak-burst**, respectively) are optional. If Bc is not specified, Bc (in bytes) will default to CIR / 32, or 1500 bytes, whichever is higher. If Be is not specified, Be (in bytes) will default to PIR / 32, or 1500 bytes, whichever is higher.

The syntax for the **police** command for dual-rate class-based policing in Cisco IOS Software Release 12.4 is as follows:

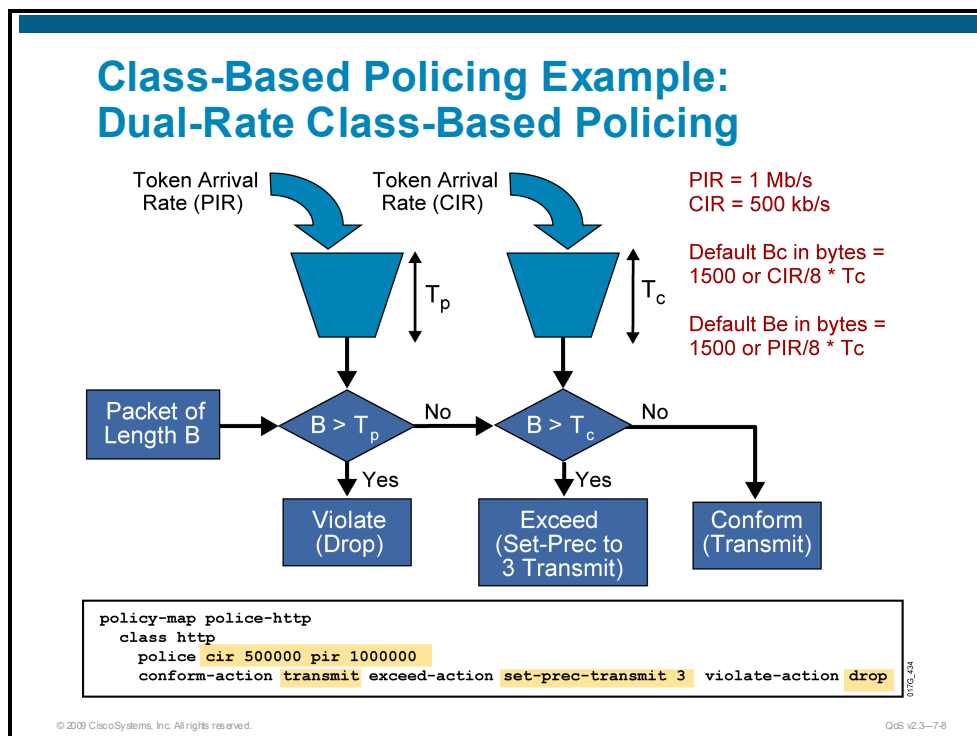
```
police cir cir [bc conform-burst] [pir pir] [be peak-burst] [conform-action action] [exceed-action action] [violate-action action]]
```


Syntax Description

Parameter	Description
cir	Committed information rate (CIR) at which the first token bucket is updated.
<i>cir</i>	Specifies the CIR value in bits per second. The value is a number from 8000 to 200,000,000.
bc	(Optional) Conform burst (bc) size used by the first token bucket for policing.
<i>conform-burst</i>	(Optional) Specifies the bc value in bytes. The value is a number from 1000 to 51,200,000.
pir	(Optional) Peak information rate (PIR) at which the second token bucket is updated.
<i>pir</i>	(Optional) Specifies the PIR value in bits per second. The value is a number from 8000 to 200,000,000.
be	(Optional) Peak burst (be) size used by the second token bucket for policing.
<i>peak-burst</i>	(Optional) Specifies the peak burst (be) size in bytes. The size varies according to the interface and platform in use.
conform-action	(Optional) Action to take on packets that conform to the CIR and PIR.
exceed-action	(Optional) Action to take on packets that conform to the PIR but not the CIR.
violate-action	(Optional) Action to take on packets exceed the PIR.

Parameter	Description
<i>action</i>	(Optional) Action to take on packets. Specify one of the following keywords: ■ drop: Drops the packet. ■ set-clp-transmit: Sets the ATM Cell Loss Priority (CLP) bit from 0 to 1 on the ATM cell and sends the packet with the ATM CLP bit set to 1. ■ set-cos-inner-transmit value: Sets the inner class of service field as a policing action for a bridged frame on the Enhanced FlexWAN module, and when using bridging features on SPAs with the Cisco 7600 SIP-200 and Cisco 7600 SIP-400 on the Cisco 7600 series router. ■ set-dscp-transmit new-dscp: Sets the IP differentiated services code point (DSCP) value and sends the packet with the new IP DSCP value setting. ■ set-dscp-tunnel-transmit value: Sets the DSCP value (0 to 63) in the tunnel header of a Layer 2 Tunnel Protocol Version 3 (L2TPv3) or Generic Routing Encapsulation (GRE) tunneled packet for tunnel marking and transmits the packet with the new value. ■ set-frde-transmit: Sets the Frame Relay discard eligible (DE) bit from 0 to 1 on the Frame Relay frame and sends the packet with the DE bit set to 1. ■ set-mpls-exp-transmit: Sets the Multiprotocol Label Switching (MPLS) experimental bits from 0 to 7 and sends the packet with the new MPLS experimental bit value setting. ■ set-prec-transmit new-prec: Sets the IP precedence and sends the packet with the new IP precedence value setting. ■ set-prec-tunnel-transmit value: Sets the precedence value (0 to 7) in the tunnel header of an L2TPv3 or GRE tunneled packet for tunnel marking and transmits the packet with the new value. ■ set-qos-transmit new-qos: Sets the quality of service (QoS) group value and sends the packet with the new QoS group value setting. ■ transmit: Sends the packet with no alteration.

Example: Dual-Rate Class-Based Policing



In the example, dual-rate policing is configured on the HTTP traffic class to limit HTTP traffic to an average committed rate of 500 kb/s and a peak rate of 1 Mb/s.

Conforming traffic will be sent as-is, traffic exceeding 500 kb/s (but not exceeding 1 Mb/s) will be marked with IP Precedence 3 and then sent, and all traffic exceeding 1 Mb/s will be dropped.

Because the Bc and Be are not specified, the default Bc in bytes will be CIR / 32 ($T_c = 0.25$ seconds), or 1500 bytes, whichever is higher. The default Be in bytes will be PIR / 32 ($T_c = 0.25$ seconds), or 1500 bytes, whichever is higher.

In this example, the default Bc = 500,000 / 32 = 15,625 bytes, and the default Be = 1,000,000 / 32 = 31,250 bytes.

The **show policy-map** command can be used to display the default settings of the Bc and Be parameters, as follows:

```
Router #show policy-map police-http
Policy Map police-http
Class silver
  police cir 500000 bc 15625 pir 1000000 be 31250
  conform-action transmit
  exceed-action set-prec-transmit 3
  violate-action drop
```

Configuring Percentage-Based Class-Based Policing

This topic describes the Cisco IOS commands that are required to configure percentage-based class-based policing.

Configuring Percentage-Based Class-Based Policing

```
router (config-pmap-c) #
```

```
police cir percent percent [bc conform-burst-in-msec] [pir  
percent percent] [be peak-burst-in-msec] [conform-action  
action] [exceed-action action] [violate-action action]
```

- Enables the same policy map for multiple interfaces with different bandwidths
- Available in Cisco IOS Software Release 12.2(13)T

```
policy-map police1  
class bulk-ftp  
  police cir percent 20 pir percent 40  
  conform-action set-dscp-transmit af11 exceed-action set-dscp-transmit 0  
  violate-action drop  
!  
interface Ethernet 0/0  
  service-policy input police1  
!  
interface Serial 0/0  
  service-policy input police1
```

© 2010 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.9

The percentage-based policing feature was introduced in Cisco IOS Software Release 12.2(13)T. Before this feature, traffic policing was configured on the basis of a user-specified amount of bandwidth available on the interface. Policy maps were then configured on the basis of that specific amount of bandwidth, meaning that separate policy maps were required for each interface with a different bandwidth. The percentage-based policing feature provides the ability to configure traffic policing based on a percentage of bandwidth available on the interface. Configuring traffic policing and traffic shaping in this manner enables customers to use the same policy map for multiple interfaces with differing amounts of bandwidth.

Configuring traffic policing based on a percentage of bandwidth is accomplished by using the **police** command.

The percentage-based policing feature also provides the option of specifying burst sizes in ms instead of in bytes when configuring traffic policing based on a percentage of bandwidth.

In the dual-rate percentage-based policing example, the bulk-ftp traffic class is policed to a CIR of 20 percent and a PIR of 40 percent of the interface bandwidth.

Because Bc and Be are not specified, the default Bc in bytes will be CIR / 32, or 1500 bytes, whichever is higher. The default Be in bytes will be PIR / 32, or 1500 bytes, whichever is higher.

Example: Configuring Percentage-Based Class-Based Policing

In the example, the CIR is 20 percent and the PIR is 40 percent (with no Bc and Be specified) and the policy is applied to an Ethernet (10 Mb/s) interface. The CIR and PIR in bits per second for the Ethernet interface are computed as follows:

$$\text{CIR} = 10 \text{ Mb/s} * 0.20 = 2 \text{ Mb/s}$$

$$\text{PIR} = 10 \text{ Mb/s} * 0.40 = 4 \text{ Mb/s}$$

The default values of the Bc and Be parameters will automatically be set to these settings:

$$\text{Bc} = 2 \text{ Mb/s} / 32 = 62,500 \text{ bytes}$$

$$\text{Be} = 4 \text{ Mb/s} / 32 = 125,000 \text{ bytes}$$

Monitoring Class-Based Policing

This topic describes the Cisco IOS commands that are used to monitor class-based policing.

Monitoring Class-Based Policing

```
router#show policy-map interface Ethernet 0/0

Ethernet0/0
Service-policy input: police1
Class-map: bulk-ftp (match-all)
 0 packets, 0 bytes
 5 minute offered rate 0 bps, drop rate 0 bps
Match: protocol ftp
  police:
    cir 20 %
    cir 2000000 bps, bc 62500 bytes
    pir 40 %
    pir 4000000 bps, be 125000 bytes
  conformed 0 packets, 0 bytes; actions:
    set-dscp-transmit cs1
  exceeded 0 packets, 0 bytes; actions:
    set-dscp-transmit default
  violated 0 packets, 0 bytes; actions:
    drop
  conformed 0 bps, exceed 0 bps, violate 0 bps
...
```

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.10

Use the **show policy-map interface** command to examine the policy map that is being applied to an interface. The output displays the percentage-based policing policy map example shown in the slide.

The output displays the actual value, calculated by Cisco IOS Software, of CIR and PIR in b/s and the Bc and Be values in bytes, based on the interface bandwidth and the percentage configurations.

Traffic Policing and Shaping for IPv6

- Similar to traffic policing and shaping for IPv4 packets
- Uses same commands used to configure policing and shaping for IPv4
- Police command enhanced to mark both IPv4 and IPv6 packets when the following keywords are used with the conform-action, exceed-action, or violate-action options:
 - **set-dscp-transmit**
 - **set-prec-transmit**

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--7-11

Traffic policing and shaping for IPv6 are similar to their implementations for IPv4 packets, and the commands used to configure traffic policing and shaping for IPv6 environments are the same commands as those used for IPv4. Traffic shaping uses flow-based queuing by default. CBWFQ can be used to classify and prioritize the packets. Class-based policing and generic traffic shaping or Frame Relay traffic shaping (FRTS) can be used for conditioning and policing traffic. Although no changes to existing configuration or command usage for policing are required for use in IPv6 environments, the **police** command has been enhanced to mark both IPv4 and IPv6 packets when the following options are used with the **conform-action**, **exceed-action**, or **violate-action** keywords:

- **set-dscp-transmit**
- **set-prec-transmit**

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Class-based policing is configured using the MQC method and is used to rate-limit a traffic class to a configured bit rate.
- Class-based policing can use a single or dual token bucket metering scheme. If a violate action is configured, a dual token bucket will be used.
- Class-based policing supports single- or dual-rate metering. Dual-rate metering allows metering of traffic based on two rates (the PIR and the CIR).
- Class-based policing supports single actions or multiactions. Multiaction allows the marking of Layer 2 and Layer 3 information at the same time.
- Cisco IOS Software can automatically calculate an optimal value of Bc and Be based on the configured policed rate.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7-12

Configuring Class-Based Shaping

Overview

Traffic shaping allows you to control outgoing traffic on an interface to match the speed of transmission to the speed of the remote interface, and to ensure that the traffic conforms to administrative quality of service (QoS) policies. You can shape traffic adhering to a particular profile to meet downstream requirements, thereby eliminating bottlenecks due to data-rate mismatches.

This lesson describes the tasks that are used to configure class-based traffic shaping in order to rate-limit certain traffic classes.

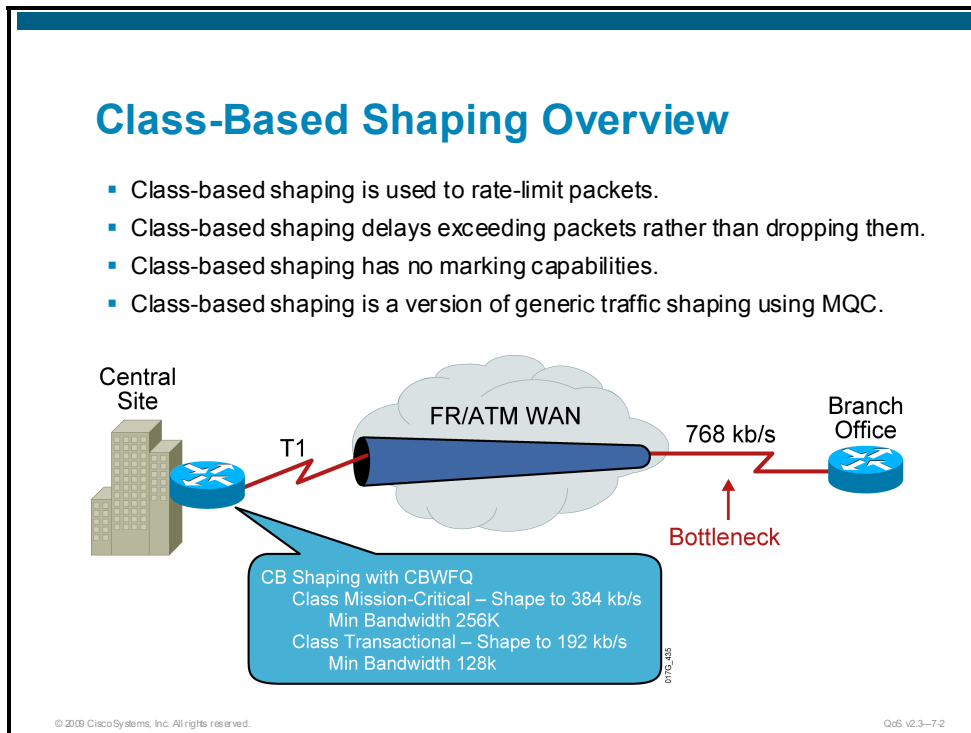
Objectives

Upon completing this lesson, you will be able to configure class-based shaping to rate-limit traffic. This ability includes being able to meet these objectives:

- Explain the key features of class-based shaping
- Explain how the two rate limits—average rate and peak rate—can be used to rate-limit traffic
- Identify the Cisco IOS Software commands required to configure class-based shaping
- Identify the Cisco IOS Software commands used to monitor class-based shaping

Class-Based Shaping Overview

This topic describes the key features of class-based shaping.



Traffic shaping allows you to control the traffic going out from an interface in order to match its transmission speed to the speed of the remote, target interface, and to ensure that the traffic conforms to policies contracted for it. You can shape traffic adhering to a particular profile to meet downstream requirements, thereby eliminating bottlenecks in topologies with traffic-rate mismatches or oversubscriptions. Class-based shaping has these properties:

- Class-based shaping, a version of generic traffic shaping, is configured via MQC.
- Class-based shaping has no packet-marking capabilities.
- Class-based shaping works by queuing exceeding packets until the packets conform to the configured shaped rate.
- Class-based shaping can also be used within the class-based weighted fair queuing (CBWFQ) queuing system. When operating within CBWFQ, the shaped rate provides an upper bandwidth limit while the bandwidth statement within CBWFQ provides a minimum bandwidth guarantee.

Traffic Shaping Methods

This topic describes how the two rate limits—average rate and peak rate—can be used to rate-limit traffic.

Traffic Shaping Methods

- Class-based shaping has two shaping methods:
 - Shaping to the **configured average** rate
 - Shaping to the **peak** rate
- Shaping to the **average** rate means forwarding packets at the configured average rate with allowed bursting up to B_e when there are extra tokens available. This is the more common method used.
- Shaping to the **peak** rate means forwarding packets at the peak rate of up to $B_c + B_e$ of traffic at every T_c . However, traffic sent above the CIR may be dropped during network congestion. Peak-rate shaping is recommended when:
 - Network has additional bandwidth available
 - Application tolerates occasional packet loss

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.3

Class-based shaping can be configured in these two ways:

- **Shaping to the configured average rate:** Shaping to the average rate forwards up to a committed burst (B_c) of traffic at every committed time window (T_c) interval, with additional bursting capability when enough tokens are accumulated in the bucket. B_c of tokens are added to the token bucket at every T_c time interval. After the token bucket is emptied, additional bursting cannot occur until tokens are allowed to accumulate, which can occur only during periods of silence or when transmit rate is lower than average rate. After a period of low traffic activity, up to $B_c + \text{excess burst } (B_e)$ of traffic can be sent.
- **Shaping to the peak rate:** Shaping to the peak rate forwards up to $B_c + B_e$ of traffic at every T_c time interval. $B_c + B_e$ of tokens are added to the token bucket at every T_c time interval. Shaping to the peak rate sends traffic at the peak rate, which is defined as the average rate multiplied by $(1 + B_e/B_c)$. Sending packets at the peak rate may result in dropping in the WAN cloud during network congestion. Shaping to the peak rate is recommended only when the network has additional available bandwidth beyond the CIR and applications can tolerate occasional packet drops.

Average rate-shaping is the more common approach being deployed.

Configuring Class-Based Shaping

This topic describes the Cisco IOS commands that are required to configure class-based shaping.

Configuring Class-Based Shaping

```
router(config-pmap-c) #
```

```
shape {average | peak} average-bit-rate [Bc] [Be]
```

- It is recommended that you omit the Bc and Be to let Cisco IOS Software select the optimal values for these variables.

```
router(config-pmap-c) #
```

```
shape {average | peak} percent [Bc] [Be]
```

- Specifies the bandwidth percentage. Valid range is a number from 1 to 100.
- It is recommended that you omit the Bc and Be to let Cisco IOS Software select the optimal values for these variables.
- The **shape (percent)** command cannot be configured for use in nested policy maps on Cisco 7500 Series routers or below.

© 2003 Cisco Systems, Inc. All rights reserved.QoS v2.3-7.4

The **shape average** and **shape peak** commands configure average and peak shaping, respectively. The Bc and Be value in bits can be explicitly configured, or Cisco IOS Software can automatically calculate their optimal value. It is not recommended that you configure the Bc and Be in order to let the Cisco IOS algorithm determine the best Bc and Be value to use. Class-based traffic shaping uses a single token bucket with a maximum token bucket size of Bc + Be.

The **shape percent** command is often used in conjunction with the **bandwidth** and **priority** commands. The **bandwidth** and **priority** commands can be used to calculate the total amount of bandwidth available on an entity (for example, a physical interface). When the **bandwidth** and **priority** commands calculate the total amount of bandwidth available on an entity, the total bandwidth is the bandwidth on the physical interface.

The **shape max-buffers** command (not shown) specifies the maximum size of the shaping queue. The shaping queue will queue excess packets until the packets conform to the shaping rate. If the shaping queue is full, packets are tail-dropped. The default max-buffer size is 1000 packets and is usually not changed.

Note The syntax for these commands may vary by platform. For the exact syntax for your router platform, refer to the *Cisco IOS Master Command List*, Release 12.4, on Cisco.com.

Example: Average Rate, Peak Rate

Class-Based Shaping Example: Average Rate, Peak Rate

Cisco IOS Software calculated values:
Bc = Be = 8000 bits, Tc = 500 ms

Peak Rate = AvgRate * (1 + Be / Bc)
= 16000 * (1 + 8000 / 8000)
= 32000 b/s

```
class-map Shape
match protocol citrix
!
policy-map ShapeAvg
class Shape
  shape average 16000
!
policy-map ShapePeak
class Shape
  shape peak 16000
!
interface Serial0/0
service-policy output ShapeAvg
!
interface Serial0/1
service-policy output ShapePeak
```

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-7.5

The figure shows an example configuration for standalone class-based shaping (no CBWFQ). Citrix traffic is classified into the Shape class.

The Shape class is then shaped to different rates on two interfaces:

- On the Serial 0/0 interface, traffic is shaped to the average rate. The Bc and Be values are not configured, allowing Cisco IOS Software to automatically calculate their optimal values.

The Citrix traffic is shaped to the average rate of 16,000 b/s. The resulting automatically determined Be and Bc values will both be 8000 bits with a Tc of 500 ms.

- On the Serial 0/1 interface, the Citrix traffic is shaped to the peak rate. Because the Be and Bc are not specified, Cisco IOS Software automatically calculates the optimal value for Tc, Bc, and Be. The shape statement is **shape peak 16000**. The resulting automatically determined Be and Bc values will each be 8000 bits with a Tc of 500 ms. Therefore, the peak rate will be as follows:

$$\begin{aligned}\text{peak rate} &= \text{ave rate} * (1 + \text{Be} / \text{Bc}) = 16000 * (1 + 8000 / 8000) \\ &= 16000 * 2 = 32000 \text{ bps}\end{aligned}$$

Showing the policy map on the interface will display the average rate, peak rate, Bc, Be, and Tc values:

```
router#show policy-map interface s0/1
```

Serial0/0

Service-policy output: **ShapePeak**

Class-map: Shape (match-all)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: protocol citrix

Traffic Shaping

Target/Average Rate	Byte Limit	Sustain bits/int	Excess bits/int	Interval (ms)	Increment (bytes)
32000/16000	2000	8000	8000	500	2000

Adapt	Queue	Packets	Bytes	Packets	Bytes	Shaping
Active	Depth			Delayed	Delayed	Active
-	0	0	0	0	0	no

Class-map: class-default (match-any)

0 packets, 0 bytes

5 minute offered rate 0 bps, drop rate 0 bps

Match: any

Example: Class-Based Shaping with CBWFQ

Class-Based Shaping Example: Class-Based Shaping with CBWFQ

- Bandwidth statement provides a minimum bandwidth guarantee for the traffic class.
- Shape statement provides a maximum traffic rate for the traffic class, and exceed traffic is queued.

```
policy-map shape-cbwfq
!
class cust1
  shape average 384000
  bandwidth 256
!
interface Serial 0/0
  service-policy output shape-cbwfq
```

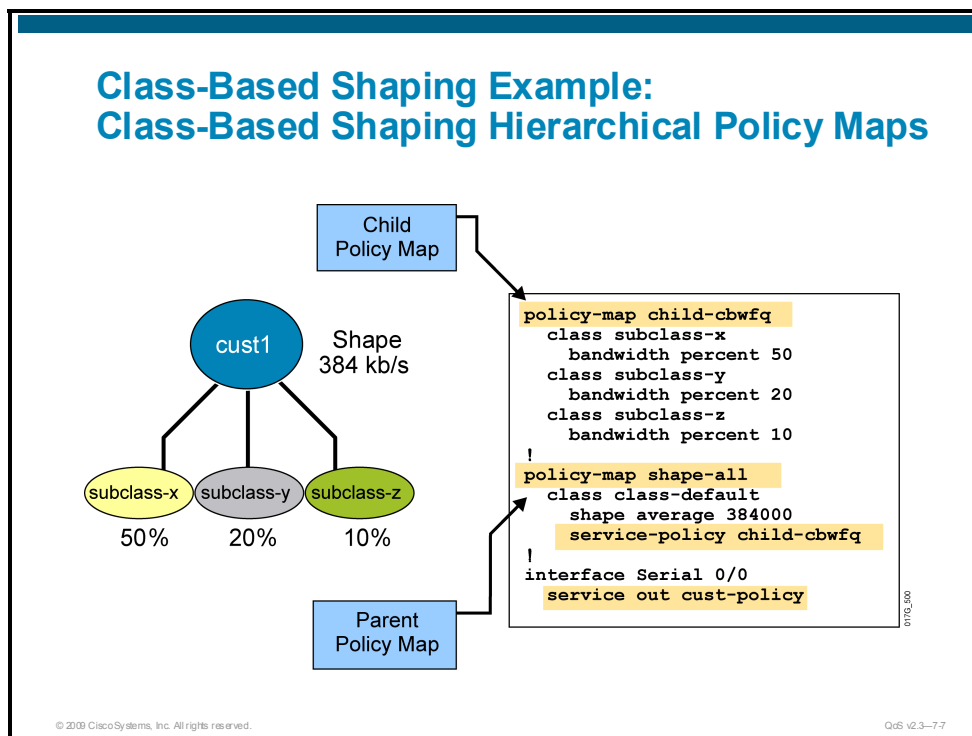
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.6

Class-based shaping can be used in combination with CBWFQ. The shape rate provides a maximum rate limit for the traffic class, while the bandwidth statement within CBWFQ provides a minimum bandwidth guarantee.

In the example, the cust1 traffic class is shaped to an average rate of 384 kb/s, and is guaranteed a minimum of 256 kb/s by the CBWFQ **bandwidth** statement. Because the Bc and Be are not specified, they will be automatically calculated by the Cisco IOS Software.

Example: Class-Based Shaping Hierarchical Policy Maps



The example uses hierarchical policy maps and configures CBWFQ inside the class-based shaping.

The parent policy is the shape-all policy. This parent policy references a child policy named child-cbwfq.

The parent policy map shape-all specifies an average shape rate of 384 kb/s for all the traffic (matched by class-default) and assigns the service policy called child-cbwfq as the child policy.

The shaped traffic is further classified into three distinct traffic subclasses:

- **subclass-x:** The subclass-x class is configured to have a minimum guarantee equal to 50 percent of the shaped bandwidth.
- **subclass-y:** The subclass-y class is configured to have a minimum guarantee equal to 20 percent of the shaped bandwidth.
- **subclass-z:** The subclass-z class is configured to have a minimum guarantee equal to 10 percent of the shaped bandwidth.

For example, in a service provider implementation, all traffic going to a specific customer edge router on the S0/0 interface is shaped, then further classified as follows:

- subclass-x class for interactive traffic
- subclass-y class for business-critical traffic
- subclass-z class for bulk traffic

The traffic going out to that specific customer edge router over the S0/0 interface is rate-limited to 384 kb/s total. The interactive traffic has a minimal guarantee of 192 kb/s (384×0.5), the business-critical traffic has a minimal guarantee of 76.8 kb/s (384×0.2), and the bulk traffic has a minimal guarantee of 38.4 kb/s (384×0.1).

Monitoring Class-Based Shaping

This topic describes the Cisco IOS commands that are used to monitor class-based shaping.

Monitoring Class-Based Shaping

```
router>
```

```
show policy-map policy-map-name
```

- Displays information about the indicated policy map, including the configuration of all classes for a specified service policy map

```
router>show policy-map shape-cbwfq
```

```
Policy Map shape-cbwfq
  Class cust1
    Traffic Shaping
      Average Rate Traffic Shaping
        CIR 384000 (bps) Max. Buffers Limit 1000 (Packets)
      Bandwidth 256 (kbps) Max Threshold 64 (packets)
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.8

The **show policy-map** command displays the policy-map configuration.

The output represents the CBWFQ in conjunction with the class-based shaping configuration example shown earlier, in which the cust1 traffic class is shaped to an average rate of 384 kb/s, with a default buffer limit of 1000 packets and a minimum bandwidth guarantee of 256 kb/s.

Monitoring Class-Based Shaping (Cont.)

```
router# show policy-map interface s0/0

Serial0/0
Service-policy output: shape-cbwfq
Class-map: cust1 (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: access-group 101
Traffic Shaping
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit  bits/int  bits/int  (ms)       (bytes)
  384000/384000    1968   7872     7872     207        984
Adapt Queue      Packets  Bytes    Packets  Bytes    Shaping
Active Depth     Delayed  Delayed  Active
-               0        0        0        0        no
Queueing
Output Queue: Conversation 266
Bandwidth 256 (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
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.9

The **show policy-map interface** command displays all service policies that are applied to the interface.

The output represents the CBWFQ in conjunction with the class-based shaping configuration example shown earlier, where the cust1 traffic class is shaped to an average rate of 384 kb/s and has a minimum bandwidth guarantee of 256 kb/s.

Among the settings shown are the class-based shaping parameters like Be, Bc, and Tc, and other statistics.

In the example, the shape average rate is 384 kb/s and the automatically calculated values for Bc and Be are 7872 bits each, with a Tc of 207 ms.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Class-based shaping delays exceeding packets rather than dropping them and has no marking capabilities.
- Class-based shaping can shape to the configured average rate or to the peak rate for packets satisfying the match criteria for a class. Peak rate = Average rate * (1 + Be / Bc).
- Class-based shaping is configured using the MQC and various shape commands.
- The **show policy-map interface** command displays the class-based traffic-shaping statistics.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--7-10

Configuring Class-Based Shaping on Frame Relay Interfaces

Overview

Frame Relay traffic shaping (FRTS) builds upon existing Frame Relay support of forward explicit congestion notification (FECN) and backward explicit congestion notification (BECN) on Cisco routers by adding new capabilities that improve the scalability and performance of a Frame Relay network. FRTS can help eliminate bottlenecks in Frame Relay networks that consist of high-speed connections at a central hub site and low-speed connections at branch spoke sites. Using FRTS, you can configure rate enforcement to either the committed information rate (CIR) or some other defined value such as the excess information rate on a permanent virtual circuit (PVC) basis. This lesson describes the tasks that are used to configure class-based traffic shaping in order to rate-limit certain traffic classes on Frame Relay interfaces.

Objectives

Upon completing this lesson, you will be able to configure class-based shaping on Frame Relay WAN interfaces to rate-limit traffic. This ability includes being able to meet these objectives:

- Explain the purpose of the Frame Relay FECN, BECN, and the DE bit
- Explain the use of FECN and BECN as a Frame Relay congestion control mechanism
- Explain how class-based shaping can adapt dynamically to available Frame Relay bandwidth by integrating BECN signals
- Explain the FECN-to-BECN propagation mechanism
- Identify the Cisco IOS commands required to configure Frame Relay adaptive class-based shaping on Frame Relay interfaces
- Identify the Cisco IOS commands used to monitor class-based shaping on Frame Relay interfaces

Frame Relay Refresher

This topic describes the purpose of the Frame Relay FECN, BECN, and the DE bit.

Frame Relay Refresher

- Frame Relay explicit congestion notification
 - FECN
 - BECN
 - Set by the Frame Relay switch to notify an end device of congestion in the network
- Implicit congestion notification
 - Provides a signal to the Frame Relay network to determine which frames to discard
 - DE bit

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.2

When the Frame Relay network becomes congested to the point that it cannot process new data transmissions, it begins to discard frames. These discarded frames are detected and then retransmitted by higher-layer protocols, thus causing more congestion. In an effort to prevent this situation, several mechanisms have been developed to notify user devices at the onset of congestion, so that the offered load may be reduced.

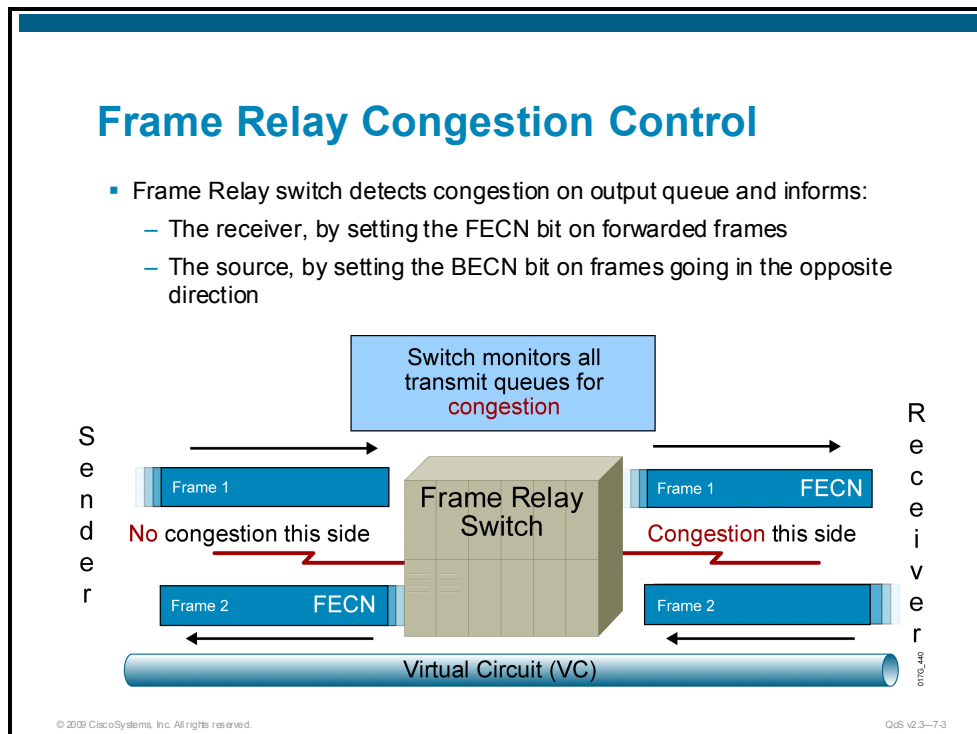
Frame Relay performs congestion notification to its Layer 2 endpoints (Frame Relay DTE) by including congestion signaling inside the Layer 2 frame headers. The FECN, BECN, and discard eligible (DE) bits in the Q.922 header of the frame provide in-band congestion signaling.

The FECN and BECN bits are set by a Frame Relay switch to notify an end device (Frame Relay DTE, which may be a router) that the end device should initiate congestion avoidance procedures.

When there is congestion, the network must decide which frames to discard. The DE bit provides the network with a signal to help determine which frames to discard. The DE bit indicates that a frame may be discarded in preference to other frames, if congestion occurs, to maintain the committed quality of service (QoS) within the network. The DE bit may be set by the user on lower-priority frames. Alternatively, the network may set the DE bit to indicate to other nodes that a frame should be preferentially selected for discard, if necessary.

Frame Relay Congestion Control

This topic describes the use of FECN and BECN as Frame Relay congestion control mechanisms.



A Frame Relay switch can explicitly report congestion in two directions: forward and backward.

When a frame queue inside a switch is congested, the switch will generate congestion signals based on the FECN and BECN bits. If congestion occurs in a queue transmitting toward the main receiver of traffic, FECN signals are sent to the receiving Layer 2 endpoint and BECN signals are sent to the sending Layer 2 endpoint.

FECN and BECN bits are not sent as separate frames, but use bits in the header of data frames.

Frame Relay Congestion Adaptation

This topic describes how class-based shaping can adapt dynamically to available Frame Relay bandwidth by integrating BECN signals.

Frame Relay Congestion Adaptation

Class-based shaping can adapt dynamically to available Frame Relay bandwidth by integrating BECN signals:

- The bit rate is **reduced** when BECN packets are received to reduce the data flow through a congested Frame Relay network.
- The bit rate is gradually **increased** when the congestion is no longer present (BECN packets are no longer received).

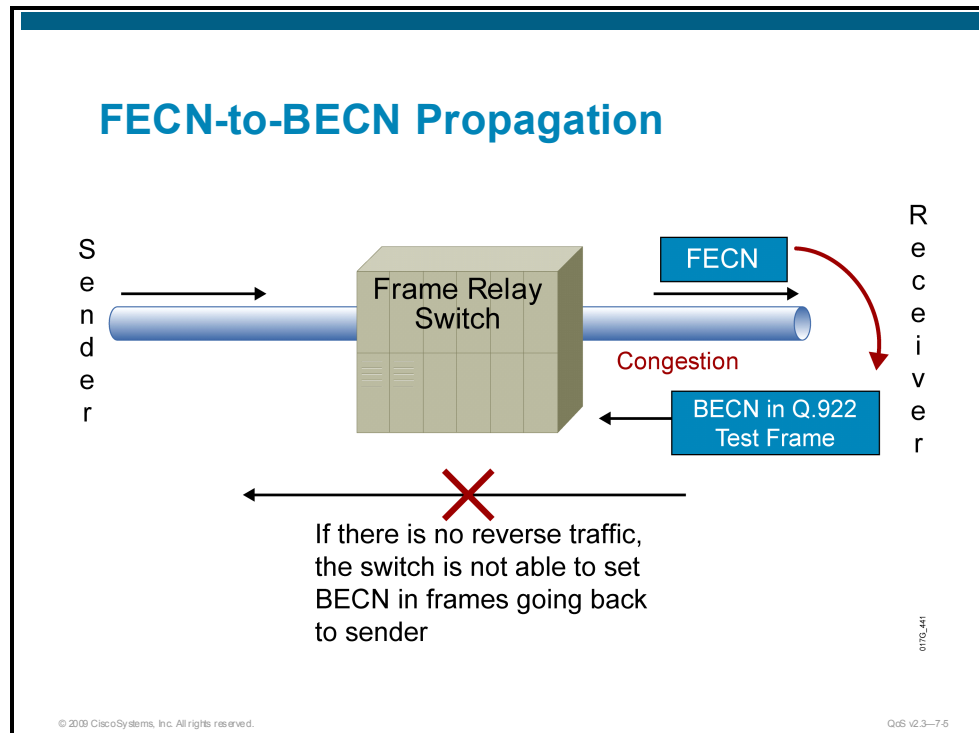
© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.4

Class-based shaping is able to respond to Layer 2 congestion in the Frame Relay network by reducing its shaping rate to three-quarters of the current rate, until the Layer 2 Frame Relay network recovers from congestion. When BECN flags are no longer received, the rate is slowly increased to the original shaping rate.

FECN-to-BECN Propagation

This topic describes the FECN-to-BECN propagation mechanism.



Another adaptation method, FECN-to-BECN propagation, configures a router Frame Relay subinterface to reflect received FECN bits as BECN in Q.922 Test Response messages. This enables the sender to notice congestion in the Layer 2 Frame Relay network, even if there is no data traffic flowing from the receiver back to the sender.

Configuring Frame Relay Adaptive Class-Based Shaping

This topic describes the Cisco IOS commands that are required to configure Frame Relay adaptive class-based shaping on Frame Relay interfaces.

Configuring Frame Relay Adaptive Class-Based Shaping

```
router(config-pmap-c) #  
shape adaptive min-rate
```

- Adapts the shaping rate when BECN bits are received.
- min-rate: Each BECN bit causes the shaping rate to be reduced to three-quarters of the previous rate but not below the min-rate.
- This command has effect only if used on Frame Relay interfaces.

```
router(config-pmap-c) #  
shape fecn-adapt
```

- Responds to FECN bits by creating test frames in the opposite direction with the BECN bit set.

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-7.6

The **shape adaptive** command configures the class-based shaping system to adapt the shaping rate to BECN indications.

The **min-rate** (also called **min-cir**) parameter specifies the minimum shaping rate that is allowed. Typically, this is set to the CIR guaranteed for the traffic class when there is congestion in the network. Additional information about BECN integration is as follows:

- If a router receives any BECNs during the current time interval, the router decreases the transmit rate by 25 percent. The router will continue to decrement with each BECN (limited to one decrement per time interval) until the traffic rate gets to min-rate, where transmit-rate ceases to decrease any further.
- After the traffic rate has decreased, it takes 16 time intervals of receiving no BECNs to start to increase traffic again. The traffic increases by $(Be + Bc) / 16$. Thus, traffic takes much longer to get back to CIR than it did to drop to min-rate (similar to slow start in TCP/IP).

The **shape fecn-adapt** command configures the class-based shaping system to respond to FECN-marked frames with BECN test frames.

Note The syntax shown in the figure for the **shape adaptive** and **shape fecn-adapt** commands is compatible with the syntax for these commands in Cisco IOS Software Release 12.4.

Example: Class-Based Shaping with Frame Relay Adaptation

MQC-Based Frame Relay Traffic Shaping Example

```
policy-map child-cbwfq
  class interactive
    bandwidth 32
  class bulk
    bandwidth 24
  !
policy-map parent-shape
  class class-default
    shape average 192000
    shape adaptive 96000
    service-policy child-cbwfq
  !
interface serial0/0
  encapsulation frame-relay
  !
interface serial0/0.1 point-to-point
  ip address 192.168.1.1 255.255.255.0
  frame-relay interface-dlci 100
  class shape
  !
  map-class frame-relay shape
    service-policy output parent-shape
```

This feature was already available for routers in the Cisco 7500 Series and above product range. In the Cisco IOS Release 12.2(13)T, it became available on lower-end platforms (Cisco 1700, 2500, 2600, 7200, and 7400 Series routers; Cisco 3620, 3631, 3640, 3660, 3725, and 3745 routers).

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-7.7

The example shows a typical MQC-based Frame Relay traffic shaping with CBWFQ configuration using hierarchical policy maps.

This is similar to hierarchical policy maps with class-based shaping. The difference is where the parent policy map with the **shape** command is applied. In class-based shaping, the parent policy map is applied to the main serial interface. With MQC-based Frame Relay traffic shaping, the parent policy map is applied to the Frame Relay PVC via a Frame Relay map-class.

If the MQC-based Frame Relay traffic-shaping feature is used, the shaping rate that was configured in the parent policy map using MQC will be used as the total available bandwidth for all traffic classes under the child policy map, if CBWFQ or low-latency queuing (LLQ) is configured. If both the **shape average** and **shape adaptive** commands are used for traffic shaping, the available bandwidth will be based on the parameters specified by the **shape adaptive** command.

In the example, all traffic out of the Frame-Relay PVC data-link connection identifier (DLCI) 100 (matched by class-default) will be shaped using the parent-shape policy map to an average rate of 192 kb/s. When the BECN bit is detected, the shape rate will start throttling down to a min-rate of 96 kb/s. The parent-shape policy map is applied to the Frame Relay map class called “shape” in this example.

In traditional (non MQC-based) FRTS configurations, all FRTS parameters such as CIR, Bc, Be, and minimum CIR (minCIR) are all defined within the Frame Relay map class.

After the aggregate traffic is shaped, the parent-shape policy map references the child-cbwfq policy map, where the two traffic classes (class interactive and class bulk) are defined with different minimum bandwidth guarantees using CBWFQ. The class interactive traffic will have a minimum bandwidth guarantee of 32 kb/s, and the class bulk traffic will have a minimum bandwidth guarantee of 24 kb/s.

Monitoring MQC-Based Frame Relay Traffic Shaping

This topic describes the Cisco IOS commands that are used to monitor MQC-based shaping.

Monitoring MQC-Based Frame Relay Traffic Shaping

```
router# show policy-map interface
Serial0/0.1: DLCI 100 -

Service-policy output: parent-shape (2811)

Class-map: class-default (match-any) (2837/0)
 16 packets, 1666 bytes
 30 second offered rate 0 bps, drop rate 0 bps
Match: any (2841)
Traffic Shaping
  Target      Byte      Sustain    Excess     Interval   Increment
Adapt
Rate          Limit  bits/int  bits/int   (ms)       (bytes)
Active
192000        1968    7872      7872       41         984      -

Queue         Packets   Bytes     Packets    Bytes      Shaping
Depth                                Delayed    Delayed    Active
0              8         833       0           0          no
....cont
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.8

The **show policy-map interface** command is used to display the parent policy shaping statistics as well as the child policy CBWFQ statistics.

Monitoring MQC-Based Frame Relay Traffic Shaping (Cont.)

```
Service-policy : child-cbwfq (2813)

  Class-map: interactive (match-all) (2815/6)
    0 packets, 0 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    Match: protocol telnet (2819)
    Weighted Fair Queuing
      Output Queue: Conversation 41
      Bandwidth 32 (kbps) Max Threshold 64 (packets)
      (pkts matched/bytes matched) 0/0
      (depth/total drops/no-buffer drops) 0/0/0

output omitted
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.9

The figure is the continuation of the previous **show policy-map interface** output.

Example: MQC-Based Frame Relay Traffic Shaping on Multipoint Main Interface

MQC-Based Frame Relay Traffic Shaping Example on Multipoint Main Interface

```
policy-map child-cbwfq
  class interactive
    bandwidth 32
  class bulk
    bandwidth 24
!
policy-map parent-shape
  class class-default
    shape average 192000
    shape adaptive 96000
    service-policy child-cbwfq

interface Serial0/1
  ip address 10.1.1.1 255.255.255.0
  encapsulation frame-relay
  frame-relay class shape
  frame-relay map ip 10.1.1.2 102 broadcast
  frame-relay map ip 10.1.1.3 103 broadcast
  frame-relay map ip 10.1.1.4 104 broadcast
!
map-class frame-relay shape
  service-policy output parent-shape
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.10

The example shows a typical MQC-based Frame Relay traffic shaping with CBWFQ configuration using hierarchical policy maps being applied to a multipoint Frame Relay main interface.

Use the **show policy-map interface** command to display the parent policy-shaping statistics as well as the child policy CBWFQ statistics.

```
router#show policy-map interface s0/1
Serial0/1: DLCI 102 -
Service-policy output: parent-shape
Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: any
Traffic Shaping
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit  bits/int  bits/int  (ms)       (bytes)
  192000/192000    1968   7872      7872     41         984
Adapt Queue      Packets  Bytes    Packets  Bytes    Shaping
Active Depth
BECN  0           0        0        0        0        Active
no
Service-policy : child-cbwfq
Class-map: interactive (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: protocol telnet
Queueing
  Output Queue: Conversation 25
  Bandwidth 32 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0
Class-map: bulk (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: protocol ftp
Queueing
  Output Queue: Conversation 26
  Bandwidth 24 (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
Serial0/1: DLCI 103 -
Service-policy output: parent-shape
Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: any
Traffic Shaping
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit  bits/int  bits/int  (ms)       (bytes)
  192000/192000    1968   7872      7872     41         984
Adapt Queue      Packets  Bytes    Packets  Bytes    Shaping
Active Depth
BECN  0           0        0        0        0        Active
no
Service-policy : child-cbwfq
Class-map: interactive (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: protocol telnet
Queueing
  Output Queue: Conversation 25
  Bandwidth 32 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0
Class-map: bulk (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
Match: protocol ftp
```

```

Queueing
  Output Queue: Conversation 26
  Bandwidth 24 (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
Serial0/1: DLCI 104 -
Service-policy output: parent-shape
Class-map: class-default (match-any)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: any
Traffic Shaping
  Target/Average   Byte   Sustain   Excess   Interval   Increment
  Rate             Limit   bits/int  bits/int  (ms)       (bytes)
  192000/192000    1968   7872     7872     41         984
Adapt Queue      Packets  Bytes    Packets  Bytes    Shaping
Active Depth
BECN  0          0        0        0        0        no
Service-policy : child-cbwfq
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: protocol telnet
Queueing
  Output Queue: Conversation 25
  Bandwidth 32 (kbps) Max Threshold 64 (packets)
  (pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0
Class-map: bulk (match-all)
  0 packets, 0 bytes
  5 minute offered rate 0 bps, drop rate 0 bps
  Match: protocol ftp
Queueing
  Output Queue: Conversation 26
  Bandwidth 24 (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

```

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- The FECN, BECN, and DE bits in the Q.922 header of the frame provide in-band congestion control.
- A Frame Relay switch detecting congestion on the output queue informs the receiver by setting the FECN bit on forwarded frames; the receiver reacts by setting the BECN bit on frames going toward the source.
- Class-based shaping can adapt dynamically to available Frame Relay bandwidth by integrating BECN signals.
- FECN-to-BECN propagation enables a router Frame Relay interface to reflect received FECN bits as BECN in Q.922 Test Response messages if no data frames are sent in the backward direction.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—7-11

Summary (Cont.)

- The **shape adaptive** command configures the class-based shaping system to adapt the shaping rate based on BECN indications.
- The **shape fecn-adapt** command configures FECN-to-BECN propagation.
- The **show policy-map interface** command displays the Frame Relay class-based shaping statistics.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3—7-12

Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation

Overview

Prior to Cisco IOS Software Release 12.2(15)T, adaptive shaping could be configured only to shape traffic rates sent to a Frame Relay cloud if congestion notifications such as backward explicit congestion notification (BECN) or forward explicit congestion notification (FECN) were received. However, in the case of voice traffic, by the time such congestion notifications were received by the sending shaper, the VoIP packet might already have been delayed or dropped. Furthermore, if fragmentation were required on slow-speed data-link connection identifiers (DLCIs), FRF.12 was enabled permanently, regardless of whether voice traffic was present.

As an alternative to congestion-adaptive traffic-shaping, Frame Relay voice-adaptive traffic shaping was introduced in Release 12.2(15)T. Frame Relay voice-adaptive traffic shaping enables adaptive shaping, a rating down to minimum committed information rate (CIR), or minCIR, and the enabling of FRF.12 fragmentation and interleaving only when voice activity is detected on the Frame Relay DLCI. Voice activity is perceived by the presence of packets in the low-latency queueing (LLQ), or the presence of H.323 signaling packets. When voice activity is no longer present, fragmentation is deactivated (after a specified interval) and the shaper resumes shaping to CIR (not minCIR). By default, Frame Relay voice-adaptive traffic shaping deactivates 30 seconds after voice activity is no longer detected, but this timeout period can be administratively tuned with the deactivation keyword and value. This lesson describes the features and functionality of Frame Relay voice-adaptive traffic shaping.

Objectives

Upon completing this lesson, you will be able to configure voice-adaptive traffic shaping and fragmentation on Frame Relay WAN interfaces. This ability includes being able to meet these objectives:

- Describe the Frame Relay voice-adaptive traffic shaping and fragmentation feature
- Describe the benefits from deploying voice-adaptive traffic shaping and fragmentation
- Describe the prerequisites for deploying voice-adaptive traffic shaping and fragmentation
- List the router platforms that support Frame Relay voice-adaptive traffic shaping
- Describe how the voice-adaptive traffic shaping and fragmentation feature works over a Frame Relay PVC
- Describe the steps in configuring voice-adaptive traffic shaping and fragmentation
- Using Cisco IOS Software CLI commands, monitor voice-adaptive traffic over a Frame Relay WAN

Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation

This topic describes Frame Relay voice-adaptive traffic shaping and fragmentation.

Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation

- Engaged during congestion
- Mechanism activation and deactivation of traffic shaping and fragmentation
- Traffic is shaped to minimum CIR
- Fragmentation per FRF.12
- When not engaged, data can burst to line speed

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.2

To address the needs of voice traffic in a Frame Relay network in which shaping is desired only when voice is present, but otherwise bursting to port-speed is desired, you can configure the Frame Relay voice-adaptive traffic shaping feature. This feature was introduced in Cisco IOS Software Release 12.2(15)T.

Frame Relay voice-adaptive traffic shaping monitors the Frame Relay permanent virtual circuit (PVC), and when voice activity is present, this feature automatically shapes the CIR to the minCIR. If no voice is present, the voice adaptive feature is turned off, allowing the data traffic to burst to the line rate, which is the CIR. The voice adaptive feature determines the presence of voice based on packets entering a low-latency queue. The feature then automatically turns traffic shaping on or off. Similarly, the Frame Relay voice-adaptive traffic shaping feature is triggered on Frame Relay Fragmentation (FRF.12) on links less than 768 kbps when voice packets are present to prevent unnecessary serialization delays.

Because Frame Relay voice-adaptive traffic shaping automatically detects the presence of voice, there could be brief quality degradation in the first couple of seconds of the first voice call made across a PVC. This occurs while the interfaces in the PVC reconfigure themselves to shape to the minimum CIR and empty out their buffers. Frame Relay voice-adaptive traffic shaping is not a predictive algorithm, and the change in behavior is triggered by voice packets flowing across the PVC or the presence of H.323 call-setup signaling.

The Frame Relay voice-adaptive traffic shaping deactivation period is tunable and, by default, is set to 30 seconds. If you tune it, you should set the timer so that the feature will not turn off frequently during normal business use (for example, between every two calls). The feature works better on PVCs that always have at least one voice call present during daytime use and relinquish shaping only at night.

Benefits in Deploying Frame Relay Voice-Adaptive Traffic Shaping

This topic describes the benefits of deploying Frame Relay voice-adaptive traffic shaping.

Benefits of Deploying Frame Relay Voice-Adaptive Traffic Shaping

- Prevents delay of voice packets during congestion
- Maximizes utilization of PVC
- Reduces router CPU utilization

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.3

Before the introduction of the Frame Relay voice-adaptive traffic shaping feature, Frame Relay adaptive shaping was used to reduce the sending rate when a network was congested. Because the adaptive shaping mechanism was triggered by network congestion, voice traffic might already have been delayed by the time the sending rate was reduced. The Frame Relay voice-adaptive traffic shaping and fragmentation feature helps to ensure voice quality by adjusting the rate of traffic based on the presence of voice on the PVC.

Frame Relay voice-adaptive traffic shaping and fragmentation provides these benefits:

- Prevents delay of voice packets when network congestion occurs by reducing the traffic rate to the minCIR and turning on fragmentation when voice packets are present on a PVC
- Maximizes utilization of the PVC by increasing the traffic rate to the CIR when voice packets are not present
- Reduces CPU utilization by turning off fragmentation when there are no voice packets present

Prerequisites for Deploying Frame Relay Voice-Adaptive Traffic Shaping

This topic describes the prerequisites for deploying Frame Relay voice-adaptive traffic shaping.

Prerequisites in Deploying Frame Relay Voice-Adaptive Traffic Shaping

- **Traffic shaping:**
 - Traffic shaping and LLQ must be configured using MQC.
- **Link fragmentation:**
 - End-to-end fragmentation must be configured in a map or on the interface.
 - LLQ must be configured.
 - End-to-end fragmentation must be configured on peer router.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3–7.4

A prerequisite for Frame Relay voice-adaptive traffic shaping is that traffic shaping and LLQ must be configured using the Modular quality of service (QoS) command-line interface (CLI), or MQC.

Prerequisites for Frame Relay voice-adaptive fragmentation are as follows:

- End-to-end fragmentation must be configured in a map class or on the interface.
- Frame Relay traffic shaping or traffic shaping using the MQC must be configured. If end-to-end fragmentation is configured on the interface, traffic shaping must be configured using the MQC.
- LLQ must be configured.
- End-to-end fragmentation must be configured on the peer router. Although the peer router may not see the fragmented packets that are expected from the router doing voice-adaptive fragmentation, the peer router will be able to handle large, unfragmented packets in addition to fragmented packets.

Note The feature supports FRF.12 fragmentation only. Neither FRF.11 Annex C nor Cisco proprietary fragmentation is supported.

Supported Platforms

This topic describes the router platforms that support Frame Relay voice-adaptive traffic shaping.

Supported Platforms

Introduced in Cisco IOS Software Release 12.2(15)T

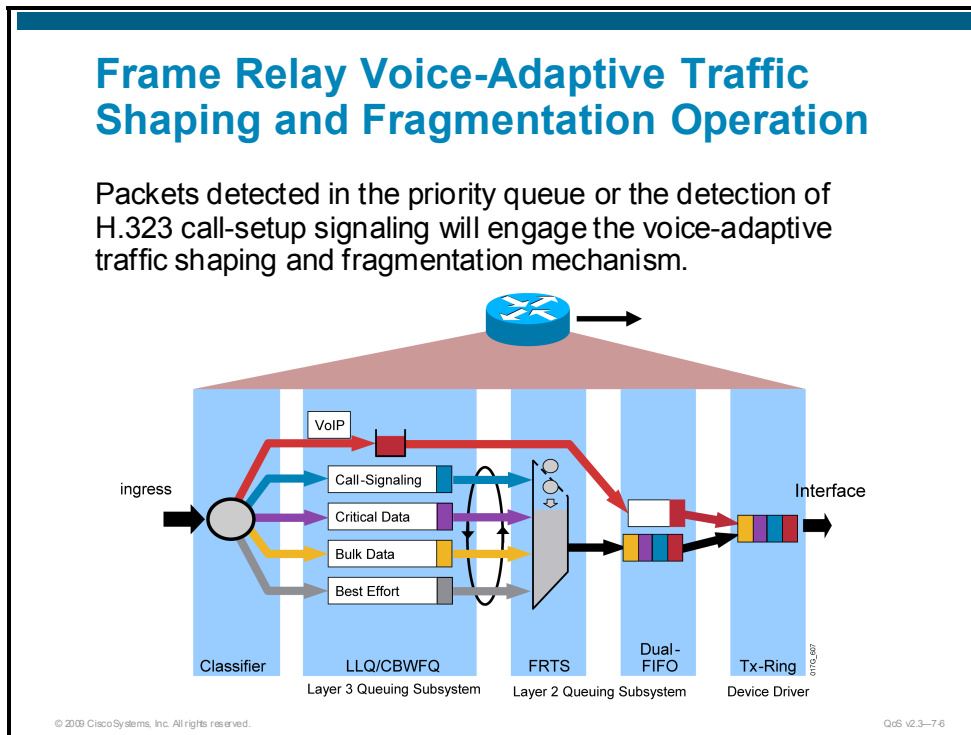
- Cisco 1700 Series Modular Access Routers
- Cisco 2600 and 3600 Series Multiservice Platforms
- Cisco 2600XM Series Multiservice Router
- Cisco 2800 and 3800 Series Integrated Services Routers
- Cisco 3700 Series Multiservice Access Routers
- Cisco 4500 Series Switches
- Cisco 7200, 7400, and 7500 Series Routers
- Not supported on Versatile Interface Processors (VIPs)

© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-7.6

The figure lists the Cisco router platforms that support Frame Relay voice-adaptive traffic shaping.

Frame Relay Voice-Adaptive Traffic Shaping and Fragmentation Operation

This topic describes how priority packets and H.323 call setup signaling packets trigger the Frame Relay voice-adaptive traffic shaping and fragmentation feature.



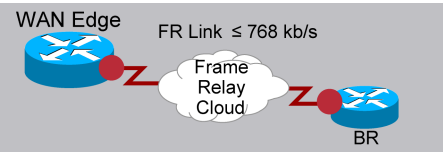
The figure shows how Frame Relay voice-adaptive traffic shaping is engaged and what occurs on the interface in order for the Frame Relay voice-adaptive traffic shaping feature to be used. After being configured, the Frame Relay voice-adaptive traffic shaping feature is automatically triggered, throttling back CIR to minCIR, if packets enter the LLQ or if H.323 call-setup signaling is detected. As packets enter the strict priority queue and the circuit is reshaped, fragmentation is engaged at Layer 2 so that priority packets are interleaved with the fragmented data packets. When the strict priority queue is void of packets, the Frame Relay circuit is reshaped back to its CIR, when data packets can burst to CIR or line speed depending on the service level agreement (SLA). Reshaping to the CIR only occurs when there are no strict priority packets in the queue, and in the absence of H.323 call-setup signaling.

Configuring the Frame Relay Voice-Adaptive Feature

This topic describes how to configure voice adaptation with end-to-end fragmentation on the interface.

Configuring Voice Adaptation with End-to-End Fragmentation on the Interface

```
interface serial0
 encapsulation frame-relay
 frame-relay fragmentation voice-adaptive deactivation 30 (FR-VATS + FRF.12)
 frame-relay interface-dlci 100
 class voice-adaptive-class (Binds voice_adaptive_class to the DLCI)
 frame-relay fragment 80 end-to-end (FRF.12 fragment for 384 kb/s PVC on the interface)
!
class-map match-all voice
 match access-group 102
class-map match-all data
 match access-group 101
!
policy-map VOICE-TRAFFIC-SHAPING
 class voice
  priority 10 (Strict Priority)
 class data
  bandwidth 10
!
policy-map TRAFFIC-SHAPE
 class class-default
  shape average 729600 3648 0 (CIR = 95% of PVC [768 kb/s], Bc to minCIR / 100, Be = 0)
  shape adaptive 364800 (Sets minimum CIR to 95% of 384 kb/s)
  shape fr-voice-adapt deactivation 30 (Enables FR-VATS [default timer])
  service-policy VOICE-TRAFFIC-SHAPING (Embedded LLQ to policy)
!
map-class frame-relay voice-adaptive-class
 service-policy output TRAFFIC-SHAPE (Embeds FR-VATS feature to map-class)
```



© 2009 Cisco Systems, Inc. All rights reserved. QoS v2.3-7.7

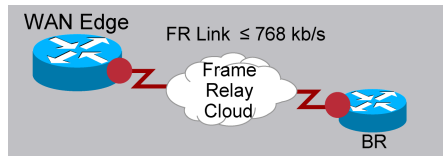
The figure shows an example of how to configure voice adaptation and end-to-end fragmentation on the interface. Notice that the strict priority queue which is used by Frame Relay voice-adaptive traffic shaping is configured within the policy map **VOICE-TRAFFIC-SHAPING** and that policy is embedded within the traffic shaper policy map **TRAFFIC-SHAPE**, where the voice-adaptive feature is enabled. In order to tie the voice-adaptive feature to the interface, the policy map **TRAFFIC-SHAPE** must be embedded into the Frame Relay map-class **voice-adaptive-class**. Data-link Connection Identifier (DLCI) 100 is bound to the voice-adaptive feature through the map-class **voice-adaptive-class**. The end-to-end fragmentation configuration can be seen on the interface, and is enabled by the command **frame-relay fragment 80 end-to-end**. The end-to-end fragmentation configuration requires that the peer router have the same end-to-end fragmentation configuration.

Configuring Voice Adaptation with End-to-End Fragmentation Through a Map-Class

```

interface serial0
 encapsulation frame-relay
 frame-relay fragmentation voice-adaptive deactivation 30 (FR-VATS + FRF.12)
 frame-relay interface-dlci 100
 class voice-adaptive-class (Binds voice_adaptive_class to the DLCI)
!
map-class frame-relay voice-adaptive-class
 frame-relay fragment 80 (FRF.12 fragment for 384 kb/s PVC through the map-class)
 service-policy output TRAFFIC-SHAPE (Embeds FR-VATS feature to map-class)
!
class-map match-all voice
 match access-group 102
class-map match-all data
 match access-group 101
!
policy-map VOICE-TRAFFIC-SHAPING
 class voice
  priority 10 (Strict Priority)
 class data
  bandwidth 10
!
policy-map TRAFFIC-SHAPE
 class class-default
  shape average 729600 3648 0 (CIR=95% of PVC [768 kb/s] Bc to minCIR / 100 Be = 0)
  shape adaptive 364800 (Sets minimum CIR to 95% of 384 kb/s)
  shape fr-voice-adapt deactivation 30 (Enables FR-VATS with default timer)
 service-policy VOICE-TRAFFIC-SHAPING (Embedded LLQ to policy)

```



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.8

The figure shows an example of how to configure the voice-adaptive feature and end-to-end fragmentation on the interface. The only difference between this figure and the previous figure is that end-to-end fragmentation on the previous figure was configured on the interface, and in this example the fragmentation is enabled by being embedded within the Frame Relay map class.

Configuring Traffic Shaping and End-to-End Fragmentation: Caveats

- Frame Relay voice-adaptive traffic shaping:
 - Can coexist with adaptive traffic shaping
 - Can be used with or without voice-adaptive fragmentation
- Frame Relay voice-adaptive fragmentation:
 - Can coexist with adaptive traffic shaping
 - Can be used with or without voice-adaptive traffic shaping
- Frame Relay voice-adaptive traffic shaping and fragmentation will respond to any packets, voice or data, in the priority queue.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.9

Voice-Adaptive Traffic Shaping and end-to-end fragmentation are designed to work together, but they can each coexist with other traffic-shaping configurations. Voice-Adaptive Traffic Shaping and end-to-end fragmentation supports only FRF.12 fragmentation. These are some caveats relative to Voice-Adaptive Traffic Shaping and end-to-end fragmentation:

- **Frame Relay voice-adaptive traffic shaping:** Frame Relay voice-adaptive traffic shaping enables a router to reduce the PVC sending rate to the minCIR when packets (usually voice packets) are detected in the low-latency queuing priority queue, or when H.323 call-setup signaling packets are present. When there are no packets in the priority queue and signaling packets are not present for a configured period of time, the router increases the PVC sending rate from minCIR to CIR to maximize throughput. Frame Relay voice-adaptive traffic shaping can be used at the same time as other types of adaptive traffic shaping. For example, if both Frame Relay voice-adaptive traffic shaping and adaptive shaping based on interface congestion are configured, the router will reduce the sending rate to minCIR if there are packets in the priority queue or the interface queue size exceeds the configured threshold. Frame Relay voice-adaptive traffic shaping can be used in conjunction with or independently of voice-adaptive fragmentation.

Note Although the priority queue is generally used for voice traffic, Frame Relay voice-adaptive traffic shaping will respond to any packets (voice or data) in the priority queue.

- **Frame Relay Voice-Adaptive End-to-End Fragmentation:** Frame Relay voice-adaptive fragmentation enables a router to fragment large data packets when packets (usually voice packets) are detected in the low-latency priority queue, or when H.323 call-setup signaling packets are present. When there are no packets in the priority queue for a configured period of time and signaling packets are not present, fragmentation is stopped. Frame Relay voice-adaptive fragmentation can be used in conjunction with or independent of Voice-Adaptive Traffic Shaping.

To use Voice-Adaptive Fragmentation, you must also have end-to-end fragmentation configured in a map class or on the interface.

Note	Although the priority queue is generally used for voice traffic, Frame Relay voice-adaptive fragmentation will respond to any packets (voice or data) in the priority queue.

Monitoring the Frame Relay Voice-Adaptive Feature

This topic describes monitoring the Frame Relay voice-adaptive traffic shaping feature using Cisco IOS Software commands.

Monitoring Frame Relay Voice-Adaptive Traffic Shaping

- `show policy map`
- `show frame-relay fragment`
- `show frame-relay pvc`
- `show policy-map interface`

© 2009 Cisco Systems, Inc. All rights reserved.

QoS V2.3--7-10

The **show policy map** command gives a static verification of the policy configuration only. No dynamic counters are included in its output. The following is a sample output:

```
WAN-EDGE#show policy TRAFFIC-SHAPE
Policy Map TRAFFIC-SHAPE
Class class-default
Traffic Shaping
  Average Rate Traffic Shaping
    CIR 729600 (bps) Max. Buffers Limit 1000 (Packets)
    Bc 3648                      Be 0
    Adapt to 364800 (bps)
    Voice Adapt Deactivation Timer 30 Sec
```

Note The highlighted section in the example shows the configuration specifics related to CIR, minCIR, and the default Frame Relay voice-adaptive traffic shaping deactivation timer.

The **show frame-relay fragment** command provides verification of the fragment size, whether static fragmentation or Voice-Adaptive Fragmentation is configured for a DLCI. Additionally, dynamic counters monitor how many frames required fragmentation in either direction. The following is a sample output:

```
WAN-EDGE#show frame-relay fragment 102
interface      dlci  frag-type  frag-size  in-frag  out-frag  dropped-frag
Serial2/0.12   102  end-to-end  480        5476     2035     0
```

Note The highlighted section shows, on the DLCI 102, the end-to-end fragmentation type, configured fragmentation size on the DLCI, and the total number of fragments sent by the designated DLCI 102.

The **show frame-relay pvc** command provides detailed traffic counters for Frame Relay DLCIs. Also, this command verifies that adaptive fragmentation has been enabled and is currently active on the DLCI. The following is a sample output:

```
WAN-EDGE#show frame-relay pvc 102
PVC Statistics for interface Serial2/0 (Frame Relay DTE)
DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial2/0.12
  input pkts 97269          output pkts 96892          in bytes 34314039
  out bytes 34622446        dropped pkts 0            in pkts dropped 0
  out pkts dropped 19        out bytes dropped 1596
  late-dropped out pkts 19  late-dropped out bytes 1596
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0           in DE pkts 0             out DE pkts 0
  out bcast pkts 62064      out bcast bytes 5315261
  5 minute input rate 150000 bits/sec, 26 packets/sec
  5 minute output rate 151000 bits/sec, 26 packets/sec
  pvc create time 3d02h, last time pvc status changed 00:13:08
  fragment type end-to-end fragment size 480 adaptive active, time left 28 secs
```

Note The highlighted section indicates that there is currently an active end-to-end session and that there are 28 seconds left before the router throttles the Frame Relay circuit back up to the CIR rate.

The **show policy interface** command is the most useful of the **show** commands, and verifies that Frame Relay voice-adaptive traffic shaping is active, due to the presence of voice traffic on the link. The following is a sample output:

```
WAN-EDGE#show policy interface serial 2/0.12
Serial2/0.12: DLCI 102 -

Service-policy output: MQC-FR-VATS-768

Class-map: class-default (match-any)
 35164 packets, 30211929 bytes
 30 second offered rate 234000 bps, drop rate 0 bps
Match: any
Traffic Shaping
  Target/Average      Byte      Sustain    Excess    Interval  Increment
  Rate               Limit  bits/int  bits/int  (ms)      (bytes)
    729600/729600    456      3648      0         5         456
Adapt Queue          Packets   Bytes     Packets   Bytes     Shaping
Active Depth                                Delayed   Delayed   Active
BECN  0              41367    30255560  13977    3645051   yes
Voice Adaptive Shaping active, time left 29 secs
```

Note The highlighted section shows that the Frame Relay voice-adaptive traffic shaping feature is active. Also, the time left for Frame Relay voice-adaptive traffic shaping to deactivate is included in the output. The output for this command has been truncated, because it is not directly relevant to this example, but if the total output were included, it would show the nested queuing policies assigned to the interface.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- When engaged, the Voice-Adaptive Traffic-Shaping feature reshapes the link to minimum CIR.
- When engaged, voice-adaptive fragmentation interleaves voice packets with data packets.
- Both voice-adaptive traffic shaping and fragmentation respond to packets in the priority queue.
- Voice-adaptive traffic shaping can coexist with adaptive traffic shaping.
- Voice-adaptive fragmentation can be configured separately from voice-adaptive traffic shaping.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--7.11

Summary (Cont.)

- Voice-adaptive traffic shaping requires the configuration of a priority queue.
- End-to-end fragmentation can be configured directly on the interface or through a map-class.
- The features were introduced with Cisco IOS Software Release 12.2(15)T .

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--7.12

Module Summary

This topic summarizes the key points discussed in this module.

Module Summary

- Traffic shaping queues excess packets to stay within the shape rate. Traffic policing typically drops excess traffic to stay within the rate limit; alternatively, it can re-mark, then send excess traffic. Both traffic shaping and policing measure traffic rate using a token bucket mathematical model.
- Class-based policing features include drop or re-mark and transmit exceeding traffic, single or dual token bucket, single- or dual-rate policing, and multiaction policing.
- Class-based shaping can shape to the average or peak rate for a traffic class.
- Class-based shaping can throttle down the traffic rate dynamically when BECN bits are received. Class-based shaping can respond to FECN bits by creating test frames in the opposite direction with the BECN bit set.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-7.1

Rate-limiting is the ability to prevent excess traffic from entering or leaving the network. Rate-limiting is required because of speed mismatches, oversubscriptions, and sub-rate access, and to prevent unwanted traffic from causing network congestion.

To measure the traffic rate, a token bucket is used. Parameters that define the operations of a token bucket include the CIR, Bc (normal burst size), Be, and Tc.

The CIR can be calculated by using the following formula:

$$\text{CIR (in b/s)} = \text{Bc (in bits)} / \text{Tc (in seconds)}$$

Both traffic policing and traffic shaping are QoS mechanisms that are used to rate-limit a traffic class. Traffic policing operates by dropping excess traffic while traffic shaping delays excess traffic with the aid of queuing.

In Cisco IOS Software, the most current rate-limiting mechanisms are class-based policing and class-based shaping. Both of these rate-limiting mechanisms are configured using the MQC. On Frame Relay interfaces, class-based shaping can be used in conjunction with CBWFQ and supports BECN adaptation and FECN-to-BECN propagation.

References

For additional information, refer to these resources:

- To learn more about configuring class-based policing, refer to *Configuring Traffic Policing* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/cnfig_traffic_polcng_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- To learn more about metering in class-based policing, refer to *RFC 2697: A Single Rate Three Color Marker* at this URL: <http://www.faqs.org/rfcs/rfc2697.html>
- To learn more about metering in class-based policing, refer to *RFC 2698: A Two Rate Three Color Marker* at this URL: <http://www.faqs.org/rfcs/rfc2698.html>
- To learn more about class-based shaping, refer to *Regulating Packet Flow on a Per-Class Basis---Using Class-Based Traffic Shaping* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/reg_pkt_flow_per_cls_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- To learn more about FRTS, refer to *MQC-Based Frame Relay Traffic Shaping* at this URL:
http://www.cisco.com/en/US/docs/ios/wan/configuration/guide/wan_mqc_fr_tfshp.html
- To learn more about FRTS, refer to *Frame Relay Traffic Shaping* at this URL:
<http://www.cisco.com/warp/public/125/21.pdf>
- To learn more about traffic policing and traffic shaping, refer to *Policing and Shaping Overview* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/polcing_shping_oview_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- For information on other traffic shaping mechanisms, refer to the software configuration documentation for your Cisco IOS Software release.

Module Self-Check

Use the questions here to review what you learned in this module. The correct answers and solutions are found in the Module Self-Check Answer Key.

- Q1) Which of the following is a major difference between traffic policing versus traffic shaping? (Source: Understanding Traffic Policing and Traffic Shaping)
- A) Traffic policing drops excess traffic while traffic shaping delays excess traffic by queuing it.
 - B) Traffic policing is applied only in the outbound direction while traffic shaping can be applied to both the inbound and outbound directions.
 - C) Traffic policing is *not* available on the Catalyst switches such as the 2950, while traffic shaping *is* available on Catalyst switches such as the 2950.
 - D) Traffic policing requires policing queues to buffer excess traffic, while traffic shaping does not require any queues to buffer excess traffic.
- Q2) Which mathematical model is used by traffic policing mechanisms to meter traffic? (Source: Understanding Traffic Policing and Traffic Shaping)
- A) token bucket
 - B) RED
 - C) FIFO metering
 - D) Predictor or Stacker
- Q3) Which three situations typically require the use of a rate-limiting mechanism? (Choose three.) (Source: Understanding Traffic Policing and Traffic Shaping)
- A) Frame Relay connection with mismatch speeds at the two end points
 - B) hub-and-spoke Frame Relay topology with oversubscription at the hub site
 - C) service provider providing sub-rate access
 - D) in the service provider high-speed backbone to avoid congestion
- Q4) When configuring single-rate class-based policing, which configuration parameter is used to enable a dual token bucket? (Source: Configuring Class-Based Policing)
- A) configuring a “violate” action
 - B) configuring an “exceed” action
 - C) configuring the PIR in addition to the CIR
 - D) configuring Be
- Q5) What is the main advantage of using multiaction policing? (Source: Configuring Class-Based Policing)
- A) to distinguish between exceeding and violating traffic
 - B) to distinguish between conforming and exceeding traffic
 - C) to allow the setting of both Layer 2 and Layer 3 QoS markers at the same time
 - D) to allow marking of the traffic before transmission

- Q6) Which two statements are true when class-based shaping is used in conjunction with CBWFQ? (Choose two.) (Source: Configuring Class-Based Shaping)
- A) The **bandwidth** command defines the minimum guaranteed bandwidth for the traffic class.
 - B) The **bandwidth** command defines the maximum guaranteed bandwidth for the traffic class.
 - C) The **shape peak** command defines the maximum rate limit for the traffic class.
 - D) The **shape average** command defines the minimum bandwidth guaranteed for the traffic class.
- Q7) Which are the two configuration options when configuring class-based traffic shaping? (Choose two.) (Source: Configuring Class-Based Shaping)
- A) shape average
 - B) shape peak
 - C) single or dual token bucket
 - D) single or multiaction traffic shaping
 - E) single- or dual-rate traffic shaping
- Q8) Which three items are features of class-based policing? (Choose three.) (Source: Configuring Class-Based Shaping)
- A) single or dual token bucket
 - B) single or multiaction policing
 - C) single- or dual-rate policing
 - D) single or dual FIFO queuing
 - E) single or dual drop threshold
- Q9) What is the min-rate? (Source: Configuring Class-Based Shaping on Frame Relay Interfaces)
- A) the minimum bandwidth guaranteed by CBWFQ for the traffic class
 - B) the minimum bandwidth guaranteed by the LLQ for the traffic class
 - C) the minimum speed the traffic rate will slow down to when the BECN bits are being received
 - D) the minimum speed the traffic rate will slow down to when the FECN bits are being received
 - E) the minimum class-based policing traffic rate for a traffic class
- Q10) What is the function of FECN-to-BECN propagation? (Source: Configuring Class-Based Shaping on Frame Relay Interfaces)
- A) to allow a receiving router to send back a test frame with the BECN bit set to the sender when the receiver has no data to send back to the sender after receiving the FECN bit
 - B) to allow a router to slow down the traffic rate to the min-rate when either the FECN or BECN bit is received
 - C) to allow a router to slow down the traffic rate to the min-rate when the BECN bit is received
 - D) to allow a router to slow down the traffic rate to the min-rate when the FECN bit is received
 - E) to allow a receiving router to set the BECN bit on a user dataframe going back to the sender after receiving the FECN bit

Module Self-Check Answer Key

- Q1) A
- Q2) A
- Q3) A, B, C
- Q4) A
- Q5) C
- Q6) A, C
- Q7) A, B
- Q8) A, B, C
- Q9) C
- Q10) A

Link Efficiency Mechanisms

Overview

In networks consisting of a large WAN covering many sites, the monthly recurring cost of even the smallest link upgrade can be too high. In some cases, the only option for low-bandwidth remote sites is to upgrade their wide-area circuit. In other cases, a set of quality of service (QoS) techniques can be used that improves the efficiency of these low-speed WAN links, which, in this context, typically refer to links of speeds less than or equal to 768 kb/s. Header compression and payload compression mechanisms reduce the size of packets, reducing delay and increasing available bandwidth on a link. Other QoS link-efficiency techniques, such as Link Fragmentation and Interleaving (LFI), allow fragile traffic types such as voice and interactive traffic to be sent either ahead or interleaved with larger, more aggressive flows. These techniques decrease latency and assist in meeting the service-level requirements of delay-sensitive traffic types such as voice.

This module describes the different link efficiency mechanisms that are available in Cisco IOS Software to implement header compression, payload compression, and LFI.

Module Objectives

Upon completing this module, you will be able to use Cisco link efficiency mechanisms to improve the bandwidth efficiency of low-speed WAN links. This ability includes being able to meet these objectives:

- Explain how link efficiency mechanisms can be used to improve bandwidth efficiency and reduce delay
- Configure class-based TCP and class-based RTP header compression to improve bandwidth efficiency and reduce delay
- Configure LFI to improve bandwidth efficiency and reduce delay

Understanding Link Efficiency Mechanisms

Overview

Interactive traffic such as Telnet and VoIP are susceptible to increased latency when network processes using large packets, such as bulk FTP, traverse WAN links. Packet delay is especially significant when FTP packets are queued on slower links (in this context, links with speeds less than or equal to 768 kb/s) within the WAN. To solve delay problems on slow bandwidth links, a method for fragmenting larger frames and then queuing the smaller frames between fragments of the larger frames is required. To meet this requirement, Cisco IOS Software supports PPP Multilink Link Fragmentation and Interleaving (MLP LFI) and Frame Relay Fragmentation (FRF.12). In addition, you can use other tools, such as header and payload compression techniques, to reduce the size of frames that are sent on WAN links.

This lesson describes different approaches for improving the efficiency of WAN links. The lesson describes link efficiency mechanisms that either compress the payload or reduce packet headers. It also describes the different Layer 2 LFI mechanisms and Frame Relay fragmentation.

Objectives

Upon completing this lesson, you will be able to explain how link efficiency mechanisms can be used to improve bandwidth efficiency and reduce delay. This ability includes being able to meet these objectives:

- Explain the various link efficiency mechanisms and their functions
- Describe the purpose of Layer 2 payload compression and explain how it affects throughput and delay
- Describe the purpose of header compression and explain how it affects throughput and delay
- Explain how VoIP packets are susceptible to increased latency when large packets such as FTP transfers traverse slow WAN links

- Explain LFI operation and how LFI reduces the delay and jitter of VoIP packets
- Identify the points in a network where link efficiency mechanisms can most effectively be employed

Link Efficiency Mechanisms Overview

This topic describes the various link efficiency mechanisms and their functions.

Link Efficiency Mechanisms Overview

- Link efficiency mechanisms are often deployed on WAN links to increase the throughput and to decrease delay and jitter.
- Cisco IOS link efficiency mechanisms include:
 - L2 payload compression (Stacker, Predictor, MPPC)
 - Header compression (TCP, RTP, class-based TCP, and class-based RTP)
 - LFI (MLP, FRF.12, and FRF.11.C)

© 2009 Cisco Systems, Inc. All rights reserved.

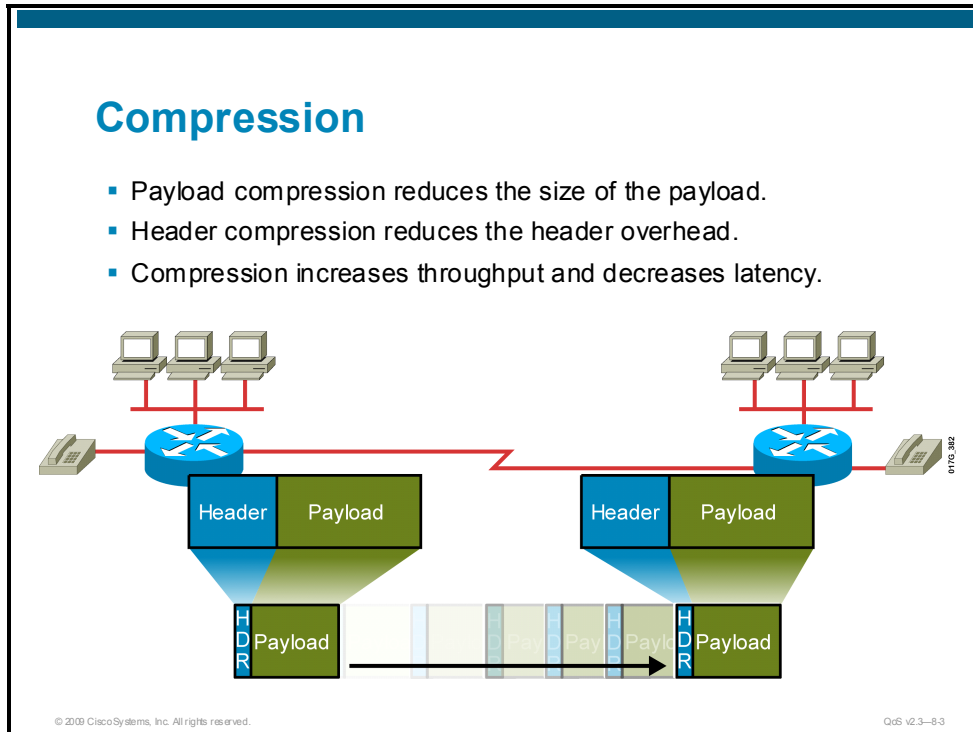
QoS v2.3-8.2

While many QoS mechanisms exist for optimizing throughput and reducing delay in network traffic, QoS mechanisms do not create bandwidth. QoS mechanisms optimize the use of existing resources, and enable the differentiation of traffic according to a policy. Link efficiency QoS mechanisms such as payload compression, header compression, and LFI are deployed on WAN links to optimize the use of WAN links.

Payload compression increases the amount of data that can be sent through a transmission resource. Payload compression is primarily performed on Layer 2 frames and therefore compresses the entire Layer 3 packet. The Layer 2 payload compression methods include Stacker, Predictor, and Microsoft Point-to-Point Compression (MPPC).

Compression methods are based on eliminating redundancy. The protocol header is an item of repeated data. The protocol header information in each packet in the same flow does not change much over the lifetime of that flow. Using header compression mechanisms, most header information can be sent only at the beginning of the session, stored in a dictionary, and then referenced in later packets by a short dictionary index. Cisco IOS header compression methods include TCP header compression, Real-Time Transport Protocol (RTP) header compression, class-based TCP header compression, and class-based RTP header compression.

LFI is a Layer 2 technique in which large frames are broken into smaller, equal-sized fragments, and transmitted over the link in an interleaved fashion with more latency-sensitive traffic flows (like VoIP). Using LFI, smaller frames are prioritized, and a mixture of fragments is sent over the link. LFI reduces the queuing delay of small frames because the frames are sent almost immediately. Link fragmentation, therefore, reduces delay and jitter by expediting the transfer of smaller frames through the hardware transmit (Tx) queue. The LFI methods available include Multilink PPP (MLP), FRF.12, and FRF.11 Annex C.



Layer 2 payload compression squeezes Layer 2 payloads (the entire Layer 3 packet). Layer 2 payload compression increases the throughput and decreases the latency in transmission, because smaller packets (with compressed payloads) take less time to transmit than the larger, uncompressed packets. Layer 2 payload compression is performed on a link-by-link basis.

Header compression methods work by *not* transmitting repeated information in packet headers throughout a session. The two peers on a PPP Layer 2 connection (a dial-up link) agree on session indices which index a dictionary of packet headers. The dictionary is built at the start of every session and is used for all subsequent (non-initial) packets. Only changing, or non-constant, parameters in the headers are actually sent along with the session index.

It is important to note that header compression is performed on a link-by-link basis. Header compression cannot be performed across multiple routers because routers need full Layer 3 header information to be able to route packets to the next hop.

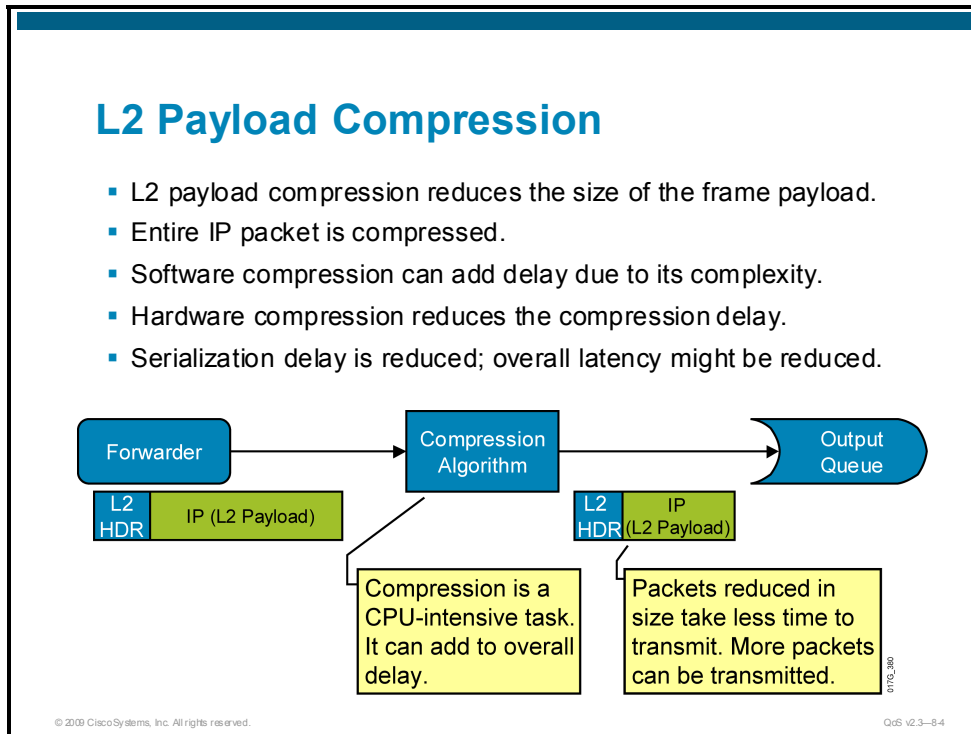
Example: Indexing Operations

Compressed RTP (cRTP) maintains the state for session contexts. A session context is defined by the combination of the IP source and destination addresses, the User Datagram Protocol (UDP) source and destination ports, and the RTP synchronization source (SSRC) field. A compressor implementation might use a hash function on these fields to index a table of stored session contexts. The compressed packet carries a small integer, called the session context identifier, to indicate in which session context that packet should be interpreted. The decompressor can use the context identifier to index its table of stored session contexts directly.

Note	Refer to RFC 2507, 2508, and 1144 for more information on header compression operations.

L2 Payload Compression

This topic describes the purpose of Layer 2 payload compression and how Layer 2 payload compression affects throughput and delay.



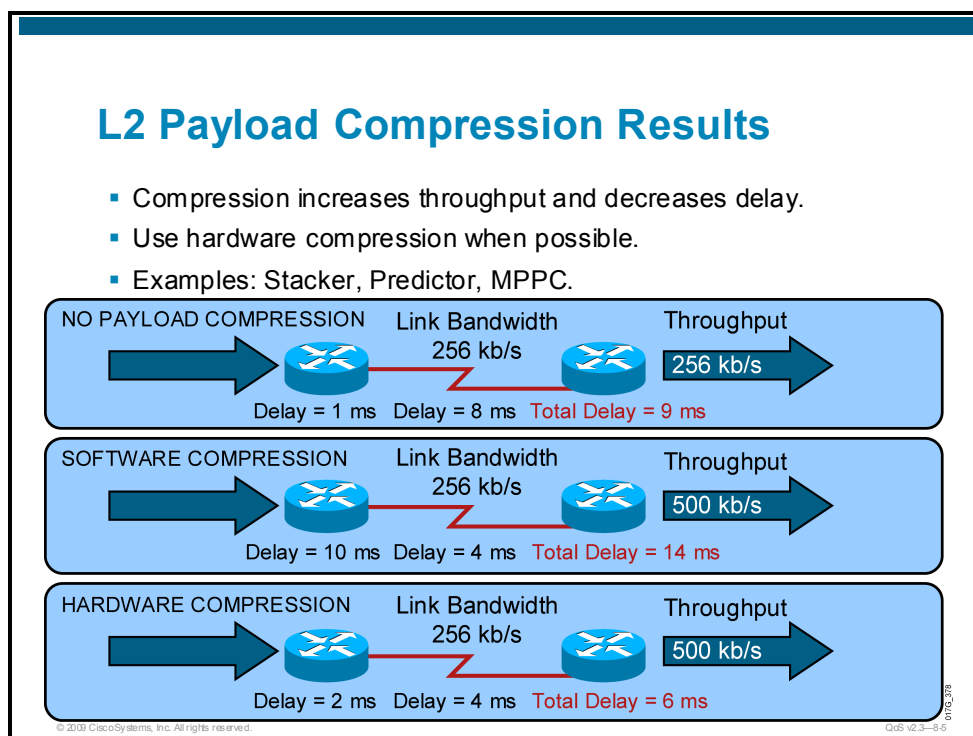
The figure shows a basic block diagram of a Layer 2 payload compression method. When a router forwards a packet, the packet is subjected to the Layer 2 compression method after it has been encapsulated at the output. The compression method squeezes the payload of the Layer 2 frame (the entire Layer 3 packet), and transmits the packet on the interface.

Layer 2 payload compression is a CPU-intensive task and can add per-packet compression delay due to the application of the compression method to each frame. The serialization delay, however, is reduced, because the resulting frame is smaller. Serialization delay is the fixed delay that is required to clock the frame onto the network interface. Depending on the complexity of the Layer 2 payload compression algorithm, overall latency might be reduced, especially on low-speed links.

Cisco routers support hardware-assisted compression to reduce the CPU load and the Layer 2 payload compression delay.

Example: L2 Payload Compression Results

The figure compares three throughput and latency scenarios on a PPP link.



If no compression is used, the throughput is limited by the link bandwidth, and the average delay is influenced by the forwarding or buffering delay, the serialization, and the propagation delay.

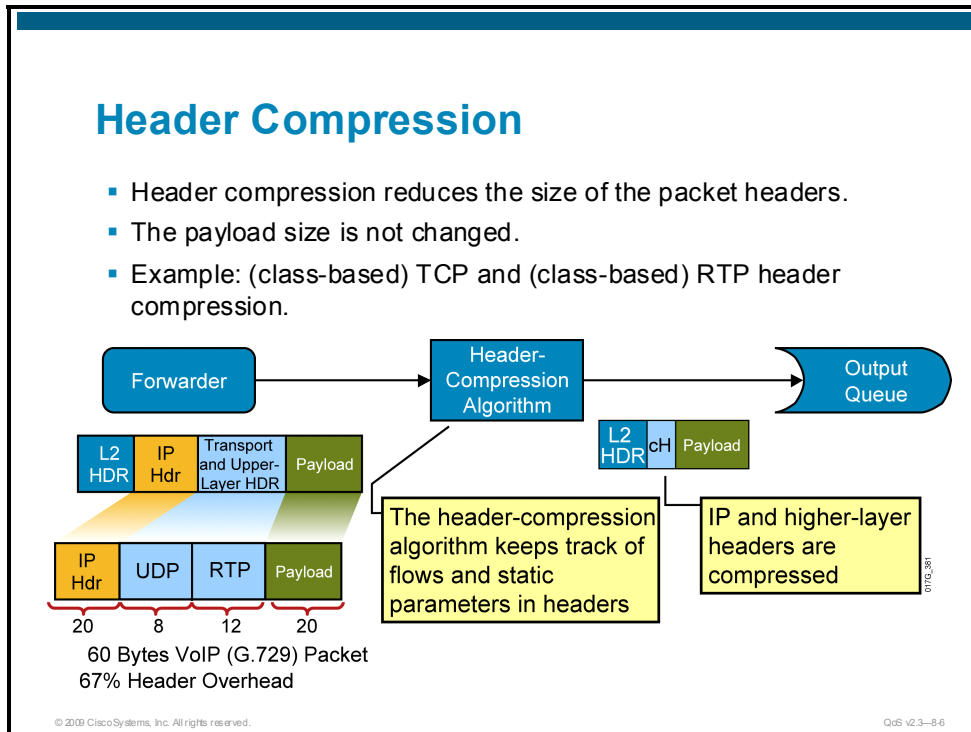
If compression is enabled—even if the serialization delay is now shorter because the frame is smaller—the compression or decompression delay may increase the overall latency between the two hops. The perceived throughput is generally increased because the size of the Layer 2 payload is reduced, therefore allowing more Layer 2 frames to be sent through a transmission resource in a given time period. The throughput is limited by the effectiveness of the Layer 2 payload compression algorithm and may be significantly higher than the link bandwidth limit.

If hardware-assisted Layer 2 payload compression is used, the compression or decompression delays may become insignificant compared to forwarding and serialization delays, and overall latency may decrease. The throughput is again limited by the effectiveness of the Layer 2 payload compression method and may be significantly higher than the link bandwidth limit.

Note Layer 2 payload compression configuration will not be covered in this module. Refer to the latest Cisco IOS documentation for configuration details.

Header Compression

This topic describes the purpose of header compression and how header compression affects throughput and delay.



Header compression increases the throughput and reduces the delay by compressing the protocol headers. Header compression is most useful for applications that generate small payloads because the protocol headers of such applications use a significant percentage of bandwidth on a link relative to their payload. Real-time applications typically generate small payloads. Target applications for header compression include Telnet and RTP applications.

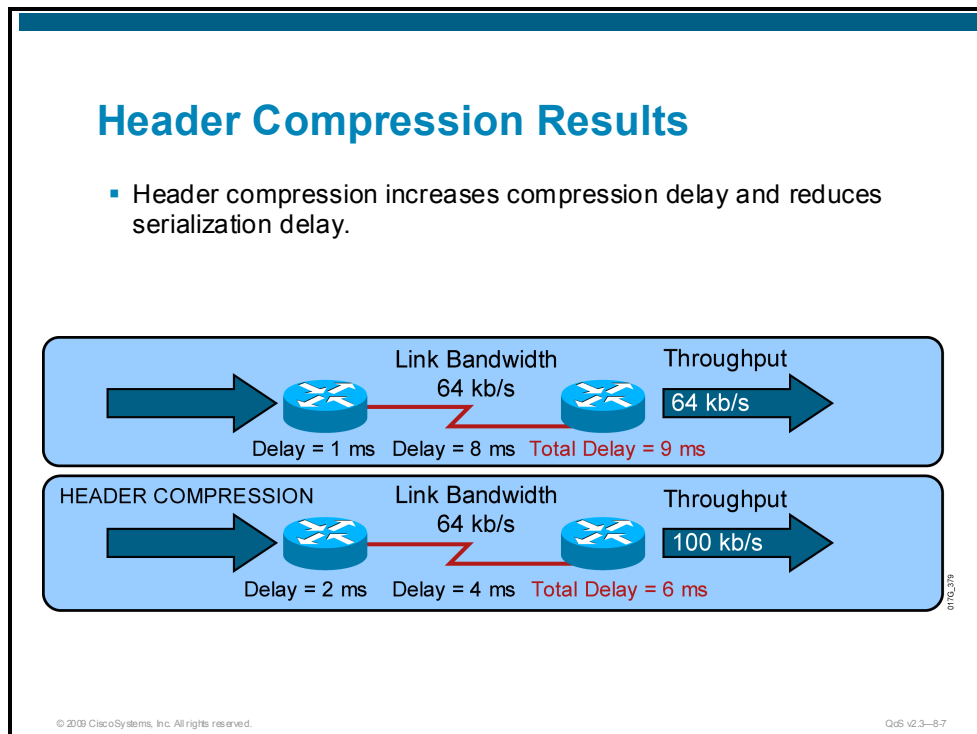
TCP and RTP header compression applies to all TCP and RTP flows. For example, if TCP compression is enabled on a link, there is no mechanism to restrict its function to specific application types. TCP header compression for bulk data transfer yields little bandwidth savings. Class-based TCP header compression can be performed only on a certain traffic class, such as the Telnet traffic class.

The header compression algorithm tracks active transport-layer connections over an interface. After the packet has been forwarded, the header compression algorithm compresses the Layer 3 and Layer 4 headers within the frame, and replaces the headers with a session index from the session dictionary (table). Only the non-constant parameters in the headers will be sent along with the session index. The packet is then sent to the output queue, and transmitted to the remote peer. When the remote peer receives the packet, the header is decompressed using the local session table, and passed to the forwarding process.

For example, without RTP header compression, the IP/UDP/RTP header overhead of the voice packet shown in the figure is about 67 percent ($40 / 60 \times 100$ percent). With RTP header compression, the IP/UDP/RTP header can be reduced to 2 or 4 bytes (without and with checksum, respectively) for most packets. Therefore, the IP/UDP/RTP header overhead can be reduced to about 9 percent ($2 / 22 \times 100$ percent) or 17 percent ($4 / 24 \times 100$ percent).

Example: Header Compression Results

The figure compares two throughput and latency scenarios on a PPP link.

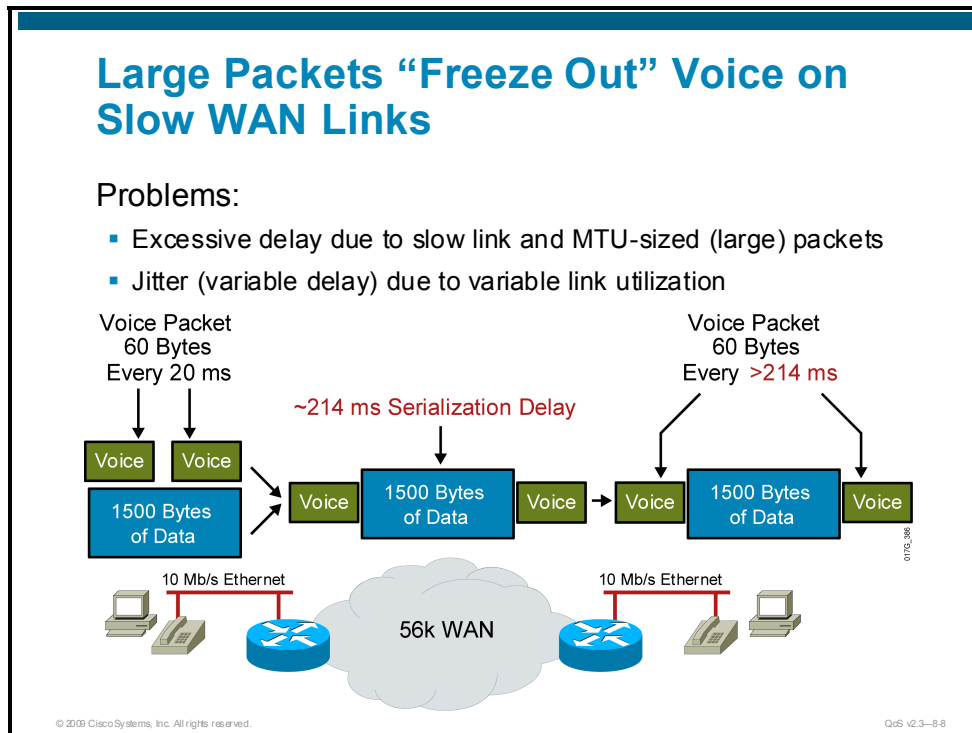


If header compression is not used, the throughput is limited by the link bandwidth, and the average delay is influenced only by the forwarding or buffering delay, the serialization, and the propagation delay.

If header compression is enabled, compressing the protocol headers causes the packet to become smaller, therefore allowing more packets to be sent through a transmission resource in a given time period to increase the throughput. Because the packet size is smaller, the serialization delay also becomes smaller, thus reducing the overall delay. Header compression has a low compression delay and a relatively low CPU overhead and is recommended on links slower than 2 Mb/s.

Large Packets “Freeze Out” Voice on Slow WAN Links

This topic describes how VoIP packets are susceptible to increased latency when large packets such as FTP transfers traverse slow WAN links.



When considering delay between two hops in a network, queuing delay in a router must be considered because it may be comparable to—or even exceed—the serialization and propagation delay on a link. In an empty network, an interactive or voice session experiences low or no queuing delay, because the session does not compete with other applications on an interface output queue. Also, the small delay does not vary enough to produce considerable jitter on the receiving side.

In a congested network, interactive data and voice applications compete in the router queue with other applications. Queuing mechanisms may prioritize voice traffic in the software queue, but the hardware queue (Tx ring) always uses a FIFO scheduling mechanism. Therefore, after packets of different applications leave the software queue, the packets will mix with other packets in the hardware queue (TxQ), even if their software queue processing was expedited. Therefore, a voice packet may be immediately sent to the hardware TxQ where two large FTP packets may still be waiting for transmission. The voice packet must wait until the FTP packets are transmitted, thus producing an unacceptable delay in the voice path. Because links are used variably, the delay varies with time and may produce unacceptable jitter in jitter-sensitive applications such as voice.

Serialization Delays

	1 Byte	64 Bytes	128 Bytes	256 Bytes	512 Bytes	1024 Bytes	1500 Bytes
56 kb/s	143 us	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kb/s	125 us	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kb/s	62.5 us	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kb/s	31 us	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kb/s	15.5 us	1 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kb/s	10 us	640 us	1.28 ms	2.56 ms	5.1 ms	10.2 ms	15 ms
1536 kb/s	5 us	320 us	640 us	1.28 ms	2.56 ms	5.12 ms	7.5 ms

0176_511

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-8

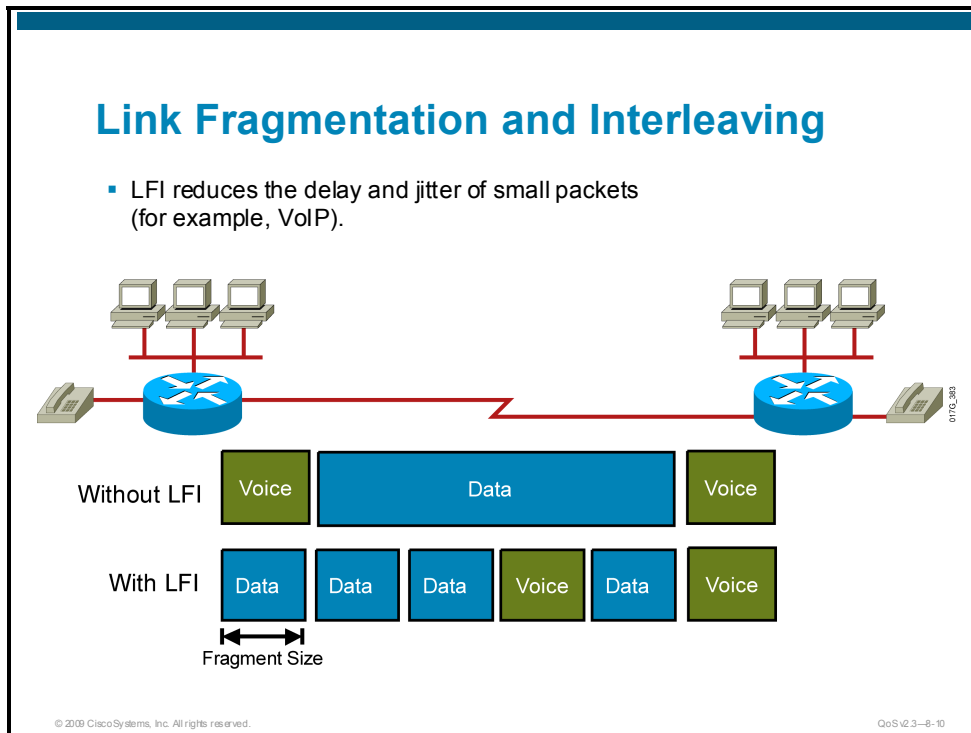
Serialization delay is the fixed delay that is required to clock a voice or data packet onto the network interface. Serialization delay is directly related to the link speed and the size of the packet.

The figure shows the serialization delay as a function of the link speed and packet size.

For example, the serialization delay for a 1500-byte packet over a 56-kb/s link will be 214 ms, while the serialization delay is only 7.5 ms over a 1.536-Mb/s link for the same 1500-byte packet.

Link Fragmentation and Interleaving

This topic describes LFI operation and how LFI reduces the delay and jitter of VoIP packets.



The use of a hybrid queuing method such as low-latency queuing (LLQ) can provide low latency and low jitter for VoIP packets, while servicing other data packets in a fair manner. But even if VoIP packets are always sent to the front of the software queue, there is still the issue of serialization delay. A large packet may be on its way out of the hardware TxQ, which uses FIFO when a VoIP packet is sent to the front of the software queue. The serialization of the large packet can cause the VoIP packet to wait for a long time before it can be transmitted out.

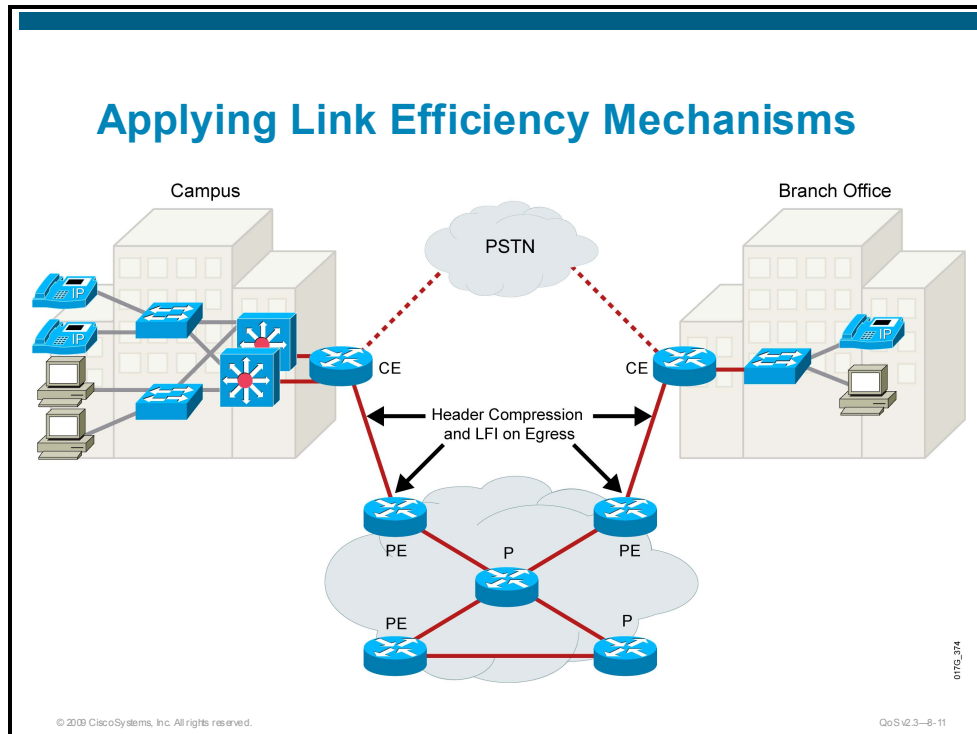
For example, the serialization delay of a 1500-byte packet over a 56-kb/s link will be 214 ms. For VoIP traffic, the maximum recommended one-way, end-to-end delay is 150 ms. Therefore, having a 1500-byte packet ahead of a VoIP packet in the hardware TxQ on a 56-kb/s link can cause the end-to-end delay of the voice packet to be over the budget of 150 ms.

The solution to this problem is to fragment the large packets so that they never cause a VoIP packet to wait for more than a predefined amount of time. The VoIP packets must also be allowed to transmit in between the fragments of the larger packets (interleaving), or there will be no point in doing the fragmenting.

When configuring the proper fragment size to use on a link, a typical goal is to have a maximum serialization delay of around 10 to 15 milliseconds. Depending on the LFI mechanisms being configured, the fragment size is either configured in bytes or in milliseconds.

Applying Link Efficiency Mechanisms

This topic describes the points in a network where link efficiency mechanisms can most effectively be employed.

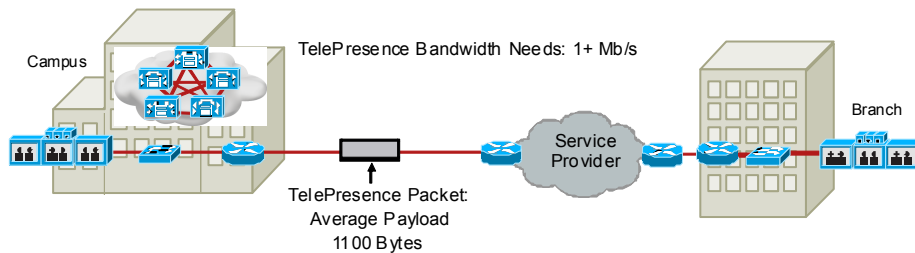


Header compression and LFI are typically configured at the WAN edge for WAN links below T1 or E1 speeds to optimize the use of the WAN link and to prevent long serialization delay.

Layer 2 payload compression is less commonly deployed on WAN links, especially without the use of hardware-assisted payload compression.

Link Efficiency Mechanisms and Cisco TelePresence

- LFI and cRTP are recommended for slow links (768 kb/s or less).
- Enabling cRTP on links carrying TelePresence is **not** recommended.
 - Large CPU impact
 - Negligible returns in bandwidth savings at TelePresence circuit speeds
- Enabling LFI on links carrying TelePresence is **not** recommended.
 - Large CPU impact
 - Unwanted jitter and out-of-order and late packets introduced by fragmentation



© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-12

Cisco TelePresence, like VoIP, is encapsulated by IP, UDP, and RTP headers. These headers, when combined, account for 40 bytes per packet (at Layer 3). To enhance bandwidth efficiency, compression tools like cRTP can reduce this overhead from 40 bytes to 2 to 5 bytes per packet. However, cRTP is the most computationally intensive QoS operation in the Cisco IOS toolset. Furthermore, it is recommended only on slow-speed links (usually 768 kb/s or less, as it is at these speeds that the bandwidth gain offsets the increased CPU cost of the operation) and it is useful only for RTP-based applications that have a small amount of payload per packet. On high-speeds links and applications like Cisco TelePresence, in which the payload of each packet averages 1100 bytes, cRTP offers no benefit and results in high CPU usage. Therefore, enabling cRTP on links carrying Cisco TelePresence is not recommended.

LFI is also not recommended on links carrying Cisco TelePresence. Like cRTP, LFI is useful only on slow-speed links. It therefore offers no benefit and results only in high CPU usage on the router. Furthermore, LFI fragments large data packets into smaller chunks. Because Cisco TelePresence packets average 1100 bytes payload per packet, LFI would fragment them. This would introduce unwanted jitter and late, out-of-order packets into the Cisco TelePresence stream.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Link efficiency mechanisms (including L2 payload compression, header compression, and LFI) deployed on WAN links can increase throughput and decrease delay and jitter.
- Payload compression uses a compression algorithm to compress the payload of Layer 2 frames.
- Header compression reduces overhead by compressing the IP and upper-layer headers.
- A VoIP packet may be sent to the hardware TxQ, where large FTP packets may still be waiting for transmission. The VoIP packet must wait until the large packets are transmitted, producing an unacceptable delay in the voice path.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--8-13

Summary (Cont.)

- LFI reduces the delay and jitter of small packets (for example, VoIP) by fragmenting the large packets to allow a VoIP packet to wait no more than a predefined amount of time.
- Header compression and LFI are typically configured at the WAN edge for WAN links below T1 or E1 speeds, to optimize the use of the WAN link and to prevent long serialization delay. Layer 2 payload compression is less commonly deployed on WAN links.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--8-14

Configuring Class-Based Header Compression

Overview

Headers exist on almost every communication layer of the OSI stack. When data is sent between workstations, headers will typically be applied at the session, transport, network, and data-link layers. Real-Time Transport Protocol (RTP) is a protocol for the transport of real-time data. RTP includes a data portion and a header portion. The data portion of RTP is a thin protocol that provides support for the real-time properties of applications, such as continuous media, including timing reconstruction, loss detection, and content identification. The header portion of RTP is considerably larger, especially when compared to the payload data that it supports. To avoid the unnecessary consumption of available bandwidth, compressed RTP (cRTP) is used on a link-by-link basis.

TCP header compression is also supported on Cisco routers to reduce the overhead that is associated with TCP and IP headers in TCP/IP packets. TCP header compression is most effective for interactive traffic with small packet size such as Telnet on slow WAN links. The modular quality of service (QoS) command-line interface (CLI), or MQC, method supports class-based RTP and TCP header compression, so that the header compression can be performed on a specific traffic class. This lesson describes class-based RTP and class-based TCP header compression, including configuration and monitoring.

Objectives

Upon completing this lesson, you will be able to configure class-based TCP and class-based RTP header compression to improve bandwidth efficiency and reduce delay. This ability includes being able to meet these objectives:

- Explain the purpose of header compression
- Calculate the overhead that you can save by using class-based TCP header compression
- Calculate the overhead that you can save on VoIP packets using class-based RTP header compression
- Identify the Cisco IOS commands required to configure class-based header compression
- Identify the Cisco IOS commands used to monitor class-based header compression

Header Compression Overview

This topic describes the purpose of header compression.

Header Compression Overview

- TCP header compression and class-based TCP header compression:
 - Compresses IP and TCP headers
 - Is used to reduce the overhead of TCP segments
 - Is most effective on slow links with many TCP sessions with small payloads (for example, Telnet)
- RTP header compression and class-based RTP header compression:
 - Compresses IP, UDP, and RTP headers
 - Is used to reduce delay and increase throughput for RTP
 - Improves voice quality
 - Is most effective on slow links
- Class-based header compression—Cisco IOS Software Release 12.2(13)T.
- Header compression is enabled on a link-by-link basis.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.2

Three methods are standardized by the Internet Engineering Task Force (IETF) for use with IP protocols:

- **TCP header compression:** Also known as the Van Jacobson or VJ header compression, TCP header compression is used to compress the packet IP and TCP headers over slow links, thus considerably improving the interactive application performance (RFC 1144).
- **RTP header compression:** Used to compress the packet IP, User Datagram Protocol (UDP), and RTP headers, thus lowering the delay for transporting real-time data, such as voice and video, over slower links (RFC 2508).
- **IP header compression:** Not discussed in this lesson. (RFC 2507).

When TCP and RTP header compression is enabled, the compression occurs by default in the fast-switched path or the Cisco Express Forwarding-switched path, depending on which switching method is enabled on the interface. Class-based header compression enables RTP or TCP header compression on a per-class basis. This feature was introduced in Cisco IOS Software Release 12.2(13)T. Decompression is not based on the class map. The receiving end will decompress all packets that come compressed from the other side. Header compression is performed on a link-by-link basis. Header compression cannot be performed across multiple routers because routers need full Layer 3 header information to be able to route packets to the next hop.

Example: RTP Header Compression

RTP header compression is often implemented on WAN links to reduce the IP, UDP, and RTP header overhead for VoIP packets. Reducing the protocol header overhead results in an increase in throughput and a decrease in serialization delay.

Class-Based TCP Header Compression

This topic describes how to calculate the approximate overhead that can be saved by using class-based TCP header compression.

Class-Based TCP Header Compression

- Most Internet applications use TCP as the transport protocol.
- Most of the information in the headers (IP and TCP) is static or predictable throughout the session.
- IP (20 bytes) and TCP (20 bytes) use 40 bytes.
- TCP header compression can squeeze these two headers into 3 to 5 bytes.
- Class-based TCP header compression allows compression on a traffic class.
- Class-based TCP header compression is configured via MQC.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.3

In TCP header compression, the IP and TCP headers, which normally use 20 bytes each, are reduced to a session index, and the variable part of the header. With all optimizations, the combined header length of 40 bytes can be reduced to a 3- to 5-byte compressed header.

After you configure a class within a policy map, class-based TCP header compression occurs on a per-class basis. Policy maps are created using the MQC. Attach the policy maps to an interface by using the **service-policy** command. The **service-policy** command gives you the option of specifying either an input service policy (for input interfaces), or an output service policy (for output interfaces). For the TCP header compression feature, only output service policies can be specified.

Enabling TCP header compression on an interface applies header compression to all TCP flows out of the interface. If TCP compression is enabled on a link, there is no mechanism to restrict its function to specific application types. TCP header compression for bulk data transfer (packets with a large payload) yields little bandwidth savings. Using the newer class-based TCP header compression, TCP header compression can be performed only on certain traffic classes; for example, you can only perform TCP header compression on the interactive traffic class where the packet payload sizes are small.

Example: Class-Based TCP Header Compression

Class-Based TCP Header Compression Example

- Link bandwidth is 64 kb/s.
- The link is used for a number of interactive TCP sessions.
- PPP encapsulation is used.
- Average packet size is 5 bytes.
- Each segment has 46 bytes of overhead (PPP, IP, and TCP headers).

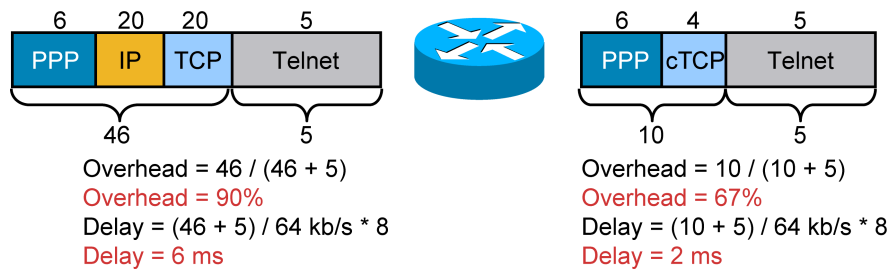
© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.4

The example shows the benefits of TCP header compression on slow links.

A 64-kb/s link is used to transport a TCP-based application using PPP as the Layer 2 framing protocol. For the case study application (Telnet), the average packet payload size is 5 bytes. Because PPP has 6 bytes of frame header, the total header overhead is $6 + 20 + 20 = 46$ bytes, including the PPP, IP, and TCP headers.

Class-Based TCP Header Compression Example (Cont.)



Payload Size	Overhead No Comp.	Delay on 64 kb/s	Overhead Comp.	Delay on 64 kb/s
10	82%	7 ms	50%	2 ms
50	48%	12 ms	17%	7 ms
100	32%	18 ms	9%	13 ms
500	8%	67 ms	2%	62 ms
1500	3%	189 ms	1%	184 ms

The figure shows the packet size before and after TCP header compression. After TCP header compression, the IP and TCP headers are reduced to 4 bytes, resulting in 10 bytes of overall headers. The overhead is reduced from 90 percent to 67 percent when small packets are used. Because of the packet size reduction, the serialization delay decreases from 6 ms to 2 ms on the same 64-kb/s link.

The table in the figure shows how TCP header compression impacts performance when different packet sizes are used. TCP header compression is most effective on small packets, and is often used on slow links.

Class-Based RTP Header Compression

This topic describes how to calculate the overhead that can be saved on VoIP packets using class-based RTP header compression.

Class-Based RTP Header Compression

- Voice sessions use **RTP**.
- RTP uses **UDP** for transport.
- Most of the information in the headers (IP, UDP, and RTP) is static throughout the session.
- IP (20 bytes), UDP (8 bytes), and RTP (12 bytes) use 40 bytes.
- RTP header compression can squeeze these three headers into 2 or 4 bytes.
- **Class-based RTP header compression allows compression on a traffic class.**
- **Class-based RTP header compression is configured via MQC.**

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.6

RTP is the standard protocol (RFC 1889) for the transport of real-time data. RTP is intended to provide end-to-end network transport functions for applications that support audio, video, or simulation data over multicast or unicast network services. RTP is used in most VoIP applications to transport voice packets.

RTP includes a data portion and a header portion. The data portion of RTP is a thin protocol that provides support for the real-time properties of applications, such as continuous media, and includes timing reconstruction, loss detection, and content identification. RTP contains a relatively large-sized header. The 12 bytes of the RTP header, combined with 20 bytes of IP header and 8 bytes of the UDP header, create a 40-byte header. For compressed-payload audio applications, the RTP packet typically has a 20- to 160-byte payload, depending on the audio compression codec. Given the size of the IP, UDP, and RTP header combinations, it is inefficient to send the header without compressing it.

To avoid unnecessary consumption of available bandwidth, the RTP header compression feature, cRTP, is used on a link-by-link basis. cRTP can reduce the header from 40 bytes to 2 or 4 bytes (4 bytes with UDP checksum enabled), which significantly reduces delay on slow links.

After you configure a class within a policy map, class-based RTP header compression occurs on a per-class basis. Policy maps are created using the MQC. Attach the policy maps to an interface by using the **service-policy** command. The **service-policy** command gives customers the option of specifying either an input service policy (for input interfaces), or an output service policy (for output interfaces). For the cRTP feature, only output service policies can be specified.

Example: Class-Based RTP Header Compression

Class-Based RTP Header Compression Example

- Link bandwidth is 64 kb/s.
- The link is used for VoIP.
- PPP encapsulation is used.
- G.729 codec is used (8 kb/s of voice data, 50 samples per second, 20 bytes per sample).
- Each segment has 46 bytes of overhead (PPP, IP, UDP, and RTP headers).

© 2009 Cisco Systems, Inc. All rights reserved.

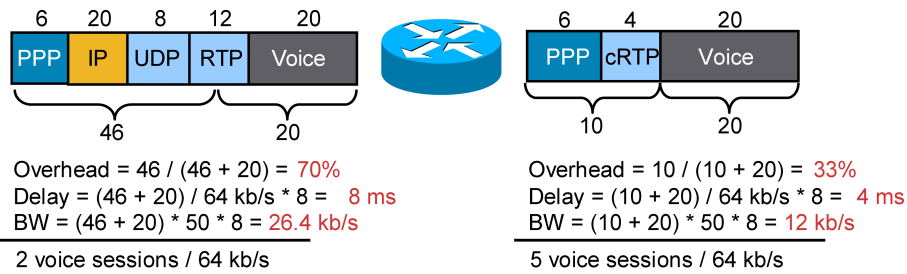
QoS v2.3-8-7

The example shows the benefits of RTP header compression on slow links.

A 64-kb/s link is used to transport VoIP using PPP as the Layer 2 framing protocol.

For the case study application (voice, using the G.729 audio compression codec), the payload size is 20 bytes. Because PPP has 6 bytes of frame header, the total header overhead is $6 + 20 + 8 + 12 = 46$ bytes, including the PPP, IP, UDP, and RTP headers.

Class-Based RTP Header Compression Example (Cont.)



Codec	Voice Bandwidth	RTP Bandwidth	RTP Overhead	cRTP Bandwidth	cRTP Overhead
G.711	64 kbps	82 kbps	22%	68 kbps	3%
G.729	8 kbps	26 kbps	61%	12 kbps	33%

The figure shows the packet size before and after RTP header compression. The IP, UDP, and RTP headers are reduced to 4 bytes, resulting in 10 bytes of overall headers. The overhead is reduced from 70 percent to 33 percent when small packets are used. Because of the packet size reduction, the serialization delay decreases from 8 ms to 4 ms, and the bandwidth that is used to transport a single voice call (using the G.729 codec) is reduced from 26.4 kb/s (66 bytes per frame * 50 frames per second * 8 bits per byte) to 12 kb/s (30 bytes per frame * 50 frames per second * 8 bits per byte). Therefore, a 64-kb/s link can support up to two G.729 voice calls without cRTP, but up to five G.729 voice calls with cRTP.

The table in the figure shows how RTP header compression impacts performance when a different audio codec is used. For the traditional G.711 voice codec, RTP header compression still optimizes its transmission over slow links. However, the difference is more obvious when using advanced, low-bandwidth codecs.

Configuring Class-Based Header Compression

This topic describes the Cisco IOS commands that are required to configure class-based header compression.

Configuring Class-Based Header Compression

```
router(config-pmap-c) #
```

```
compression header ip [rtp | tcp ]
```

- Enables RTP or TCP IP header compression for a specific traffic class.
- If the **rtp** or **tcp** option is not specified, both RTP and TCP header compressions are configured.
- The number of concurrent compressed connections is automatically determined based on interface bandwidth.
- Can be used at any level in the policy map hierarchy configured with MQC.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-6.8

Configure class-based TCP and RTP header compression within a policy map using the **compression header ip** command.

If you do not specify either RTP or TCP header compression, both RTP and TCP header compressions will be configured.

The syntax shown in the figure for the **compression header ip** command is the same as the syntax for the **compression header ip** command in Cisco IOS Software Release 12.4.

Note	Header compression is auto-negotiated only on PPP links. On Frame Relay or high-level data link control (HDLC) links, both ends of the links have to match.

Example: Configuring Class-Based TCP Header Compression

Example: Configuring Class-Based TCP Header Compression

```
class-map interactive
  match protocol telnet
!
policy-map cust1
  class interactive
    bandwidth 64
    compression header ip tcp
!
<output omitted>
!
int s0/0
  service-policy output cust1
<output omitted>
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-10

In the figure, the **compression header ip** command is configured to use TCP header compression for a traffic class called “interactive.” The “interactive” traffic class is part of a policy map called “cust1.” This “cust1” policy map is applied to the s0/0 interface in the outbound direction.

This policy provides a minimum bandwidth guarantee of 64 kb/s for the “interactive” traffic class and will perform TCP header compression on the “interactive” traffic class (all Telnet packets in this example) leaving the s0/0 interface.

Example: Configuring Class-Based RTP Header Compression

Example: Configuring Class-Based RTP Header Compression

```
class-map voip
  match protocol rtp
  !
policy-map cust1
  class voip
    priority 384
    compression header ip rtp
  !
<output omitted>
!
int s0/0
service-policy output cust1
<output omitted>
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-11

In the figure, the **compression header ip** command has been configured to use RTP header compression for a traffic class called “voip.” The voip traffic class is part of a policy map called “cust1.” This cust1 policy map is applied to the s0/0 interface in the outbound direction.

This policy provides a maximum bandwidth guarantee of 384 kb/s for the voip traffic class and will perform RTP header compression on the voip traffic class (all RTP packets in this example) leaving the s0/0 interface.

Monitoring Class-Based Header Compression

This topic describes the Cisco IOS commands that are used to monitor class-based header compression.

Monitoring Class-Based Header Compression

router>

```
show policy-map interface interface-name
```

- Displays the packet statistics of all classes configured for all service policies on the specified interface

```
router>show policy-map interface Serial 0/0
Serial0/0
Service-policy output:cust1
Class-map: voip (match-all)
1005 packets, 64320 bytes
30 second offered rate 16000 bps, drop rate 0 bps
Match:protocol rtp
Queueing
  Strict Priority
  Output Queue: Conversation 264
  Bandwidth 384 (kbps) Burst 9600 (Bytes)
  (pkts matched/bytes matched) 1000/17983
  (total drops/bytes drops) 0/0compress:
compress:
header ip rtp
UDP/RTP Compression:
Sent:1000 total, 999 compressed,
41957 bytes saved, 17983 bytes sent
3.33 efficiency improvement factor
99% hit ratio, five minute miss rate 0 misses/sec, 0 max rate 5000 bps
```

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6-12

The **show policy-map interface** command output displays the type of header compression configured (RTP in this example), the interface to which the policy map called “cust1” is attached (serial 0/0), the number of packets sent, the number of packets compressed, the number of bytes saved, and the number of bytes sent.

Other statistical information provided in the output includes the “efficiency improvement factor,” which indicates the percentage of increased bandwidth efficiency as a result of header compression. For example, an efficiency improvement factor of 3.33 means 330 percent efficiency improvement. The “hit ratio” is the percentage of packets that are found in the context database. In most instances, this percentage should be high. The “five-minute miss rate” is the number of traffic flows in the last five minutes that were *not* found in the context database. The rate is the actual traffic rate after the packets are compressed.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- TCP header compression compresses the IP and TCP headers. RTP header compression compresses the IP, UDP, and RTP headers. Header compression reduces delay.
- Class-based TCP header compression enables TCP header compression on a per-class basis. TCP header compression can reduce the header from 40 bytes to 3 to 5 bytes.
- RTP header compression feature (cRTP) can reduce the header from 40 bytes to a 2- or 4-byte header (4 bytes with UDP checksum enabled), which significantly reduces delay on slow links.
- Class-based TCP and RTP header compression is configured by using the **compression header ip** command in policy-map class configuration mode.
- The **show policy-map interface** command output displays the type and statistics of the header compression configured.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--8-13

Configuring Link Fragmentation and Interleaving

Overview

Because the hardware transmit queue (TxQ) is a FIFO queue, having a large packet such as an FTP packet in front of a small packet like a VoIP packet can cause excessive delay for the VoIP packet. Link fragmentation and interleaving (LFI) allows the large packets to be fragmented into smaller fragments, and then the small unfragmented packets are interleaved in between the fragments. LFI reduces the delay and jitter of small packets such as VoIP packets over a WAN link.

This lesson describes the configuration and monitoring of two different LFI mechanisms supported in Cisco IOS Software, Multilink PPP LFI (MLP LFI) and Frame Relay Fragmentation (FRF.12).

Objectives

Upon completing this lesson, you will be able to configure LFI to improve bandwidth efficiency and reduce delay. This ability includes being able to meet these objectives:

- Identify the different options available for link fragmentation
- Given a list of link speeds and a specific delay requirement, determine the proper fragment size to use at each link speed and the typical delay requirements for VoIP packets
- Identify the Cisco IOS commands required to configure MLP LFI
- Identify the Cisco IOS commands used to monitor MLP LFI
- Explain when FRF.12 can be used and how FRF.12 affects VoIP packets
- Identify the Cisco IOS commands required to configure FRF.12
- Identify the Cisco IOS commands required to monitor FRF.12

Fragmentation Options

This topic describes the different options that are available for link fragmentation.

Fragmentation Options

Cisco IOS Software LFI mechanisms include:

- **Multilink PPP with interleaving:** PPP links
- **FRF.12:** Frame Relay PVC carrying data traffic, including VoIP over Frame Relay traffic
- **FRF.11 Annex C:** Frame Relay PVC carrying VoFR traffic

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.2

LFI is a Layer 2 technique, in which all Layer 2 frames are broken into small, equal-sized fragments, and transmitted over a link in an interleaved fashion. LFI reduces delay and jitter by expediting transfer of smaller frames through the hardware TxQ.

There are three LFI mechanisms implemented in Cisco IOS Software:

- **MLP LFI:** By far the most common and widely used form of LFI
- **FRF.12 Frame Relay LFI:** Used with Frame Relay data connections
- **FRF.11 Annex C LFI:** Used with Voice over Frame Relay (VoFR)

Note For configuration information on FRF.11.C, refer to the latest Cisco IOS documentation.

Serialization Delay and Fragment Sizing

This topic describes the proper fragment size to use on links based on their speed, and identifies the typical delay requirements for VoIP packets.

Serialization Delays

- For 1500-byte packets, fragmentation is not necessary above T1 (1.5 Mb/s)

Link Speed	Frame Size						
	1 Byte	64 Bytes	128 Bytes	256 Bytes	512 Bytes	1024 Bytes	1500 Bytes
56 kb/s	143 us	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kb/s	125 us	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kb/s	62.5 us	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kb/s	31 us	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kb/s	15.5 us	1 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kb/s	10 us	640 us	1.28 ms	2.56 ms	5.1 ms	10.2 ms	15 ms
1536 kb/s	5 us	320 us	640 us	1.28 ms	2.56 ms	5.12 ms	7.5 ms

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-6.3

Serialization delay is the fixed delay required to clock a voice or data packet onto the network interface. Serialization delay is directly related to the link speed and the size of the packet.

The figure shows the serialization delay as a function of the link speed and packet size. For example, the serialization delay for a 1500-byte packet over a 64-kb/s link will be:

$$(1500 \text{ bytes} * 8 \text{ bits per byte}) / 64 \text{ kb/s} = 187 \text{ ms}$$

To ensure low delay and jitter for voice packets on slow links, the recommended standard goal for serialization delay is about 10 to 15 ms. Therefore, on a T1 or higher speed link, LFI is not necessary because the serialization delay for a 1500-byte packet is only 7.5 ms on a T1 link. A T1 link has a bandwidth of 1536 kb/s, excluding the T1 framing overhead.

Fragment Size Recommendation for Voice

Link Bandwidth	10 ms	20 ms	30 ms	40 ms	50 ms	100 ms	200 ms
56 kb/s	70 bytes	140 bytes	210 bytes	280 bytes	350 bytes	700 bytes	1400 bytes
64 kb/s	80 bytes	160 bytes	240 bytes	320 bytes	400 bytes	800 bytes	1600 bytes
128 kb/s	160 bytes	320 bytes	480 bytes	640 bytes	800 bytes	1600 bytes	3200 bytes
256 kb/s	320 bytes	640 bytes	960 bytes	1280 bytes	1600 bytes	3200 bytes	6400 bytes
512 kb/s	640 bytes	1280 bytes	1920 bytes	2560 bytes	3200 bytes	6400 bytes	12800 bytes
768 kb/s	1000 bytes	2000 bytes	3000 bytes	4000 bytes	5000 bytes	10000 bytes	20000 bytes
1536 kb/s	2000 bytes	4000 bytes	6000 bytes	8000 bytes	10000 bytes	20000 bytes	40000 bytes

↑
Recommendation
for Voice (< 15 ms)

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.4

To meet the standard serialization delay goal of no more than 10 to 15 ms to ensure low delay and jitter for voice packets, you should configure a fragment size of about 80 bytes per every 64 kb/s of the clocking rate for the interface.

Depending on the LFI mechanism, the fragment size is either configured in bytes or in milliseconds. For example, MLP LFI maximum fragment size is configured in milliseconds, while FRF.12 maximum fragment size is configured in bytes.

Example: Determining the Proper Fragment Size

The following are example values of fragment size, based on the rule that a fragment size of about 80 bytes per every 64 kb/s of the clocking rate should be configured:

- On a 64-kb/s link, the proper fragment size to use is 80 bytes.
- On a 128-kb/s link, the proper fragment size to use is 160 bytes (80 * 2).
- On a 192-kb/s link, the proper fragment size to use is 240 bytes (80 * 3).
- On a 256-kb/s link, the proper fragment size to use is 320 bytes (80 * 4).
- And so on.

Configuring MLP with Interleaving

This topic describes the Cisco IOS commands required to configure MLP with interleaving.

Configuring MLP with Interleaving

Configuration steps:

- Enable MLP on an interface (using a multilink group interface).
- Enable MLP interleaving on the multilink interface.
- Specify maximum fragment size by setting the maximum delay on the multilink interface.

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.5

To configure MLP with interleaving, you must perform these configuration steps:

- Step 1** Enable MLP on a PPP interface.
- Step 2** On the multilink interface, enable interleaving within MLP.
- Step 3** In the multilink interface configuration, specify the maximum fragment size by specifying the maximum desired serialization delay in ms.

Configuring MLP with Interleaving

```
router(config-if)#
```

```
ppp multilink
```

- Enables MLP.

```
router(config-if)#
```

```
ppp multilink interleave
```

- Enables interleaving of frames with fragments.

```
router(config-if)#
```

```
ppp multilink fragment delay delay
```

- Configures maximum fragment delay in ms.
- The router calculates the maximum fragment size from the interface bandwidth and the maximum fragment delay.
- Fragment size = interface bandwidth * maximum fragment delay.
- Default maximum fragment delay is 30 ms.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3-8.6

The **ppp multilink** command enables MLP on a PPP interface.

The **ppp multilink interleave** command enables interleaving of fragments within the multilink connection.

The **ppp multilink fragment delay delay** command specifies the maximum desired fragment delay for the interleaved multilink connection. The maximum fragment size is calculated from the interface bandwidth and the specified maximum delay. The default is set at 30 ms. To support voice packets, a maximum fragment size of 10 to 15 ms should be used.

If distributed Cisco Express Forwarding is configured on a Versatile Interface Processor (VIP) interface, MLP with interleaving will run in distributed mode on the VIP.

The syntax shown in the figure for the **ppp multilink** command is compatible with the syntax for the **ppp multilink** command in Cisco IOS Software Release 12.4. The syntax shown in the figure for the **ppp multilink interleave** command is the same as the syntax for the **ppp multilink interleave** command in Cisco IOS Software Release 12.4.

The syntax for the **ppp multilink fragment delay** command in Cisco IOS Software Release 12.4 is as follows:

```
ppp multilink fragment delay milliseconds [microseconds]
```

Syntax Description

Parameter	Description
<i>milliseconds</i>	Maximum amount of time, in milliseconds, that should be required to transmit a fragment. Valid values range from 0 to 1000 milliseconds. The default is 30 milliseconds. If the desired delay should be in microseconds, set the milliseconds argument to 0 and enter a value for the <i>microseconds</i> argument.
<i>microseconds</i>	(Optional) Maximum amount of time, in microseconds, that should be required to transmit a fragment. Valid values range from 1 to 999 microseconds.

Example: MLP with Interleaving

MLP with Interleaving Example

```
interface Multilink1
ip address 172.22.130.1 255.255.255.252
ppp multilink
ppp multilink group 1
ppp multilink fragment delay 10
ppp multilink interleave
bandwidth 128
service-policy output llq-policy
!
interface Serial0/0
no ip address
encapsulation ppp
ppp multilink
ppp multilink group 1
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.7

The figure shows an example configuration of MLP with interleaving on a multilink group interface.

A non-default maximum desired delay of 10 ms is configured.

Monitoring MLP with Interleaving

This topic describes the Cisco IOS commands that are used to monitor MLP with interleaving.

Monitoring MLP Interleaving

router>

```
show interfaces multilink interface-number
```

- Displays MLP statistics including the number of interleaved frames

```
router>show interfaces multilink 1
Multilink1 is up, line protocol is up
Hardware is multilink group interface
Internet address is 172.22.130.1/30
MTU 1500 bytes, BW 64 Kbit, DLY 1000000 usec,
    reliability 255/255, txload 27/255, rxload 1/255
Encapsulation PPP, loopback not set
Keepalive set (10 sec)
DTR is pulsed for 2 seconds on reset
LCP Open, multilink Open
Open: IPCP
Last input 00:00:03, output never, output hang never
Last clearing of "show interface" counters 6d00h
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0/2441 (size/max total/threshold/drops/interleaves)
Conversations 0/7/16 (active/max active/max total)
Reserved Conversations 0/0 (allocated/max allocated)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 7000 bits/sec, 6 packets/sec
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS 2.3-8.8

The **show interfaces multilink** command output includes MLP LFI statistics information and indicates whether MLP interleaving is enabled on the interface. Multilink should be in the open state along with Link Control Protocol (LCP) and IP Control Protocol (IPCP).

Monitoring MLP Interleaving (Cont.)

router#

```
debug ppp multilink fragments
```

- Displays information about individual multilink fragments and interleaving events

```
router# debug ppp multilink fragments
Multilink fragments debugging is on

Mar 17 20:03:08.995: Se0/0 MLP-FS: I seq C0004264 size 70
Mar 17 20:03:09.015: Se0/0 MLP-FS: I seq 80004265 size 160
Mar 17 20:03:09.035: Se0/0 MLP-FS: I seq 4266 size 160
Mar 17 20:03:09.075: Se0/0 MLP-FS: I seq 4267 size 160
Mar 17 20:03:09.079: Se0/0 MLP-FS: I seq 40004268 size 54
Mar 17 20:03:09.091: Se0/0 MLP-FS: I seq C0004269 size 70
Mar 17 20:03:09.099: Se0/0 MLP-FS: I seq C000426A size 70
Mar 17 20:03:09.103: Mu1 MLP: Packet interleaved from queue 24
Mar 17 20:03:09.107: Se0/0 MLP-FS: I seq C000426B size 70
Mar 17 20:03:09.119: Se0/0 MLP-FS: I seq C000426C size 70
Mar 17 20:03:09.123: Mu1 MLP: Packet interleaved from queue 24
Mar 17 20:03:09.131: Mu1 MLP: Packet interleaved from queue 24
Mar 17 20:03:09.135: Se0/0 MLP-FS: I seq C000426D size 70
Mar 17 20:03:09.155: Se0/0 MLP-FS: I seq C000426E size 70
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8.9

The **debug ppp multilink fragments** command is a valuable troubleshooting tool when monitoring MLP LFI operations. This command outputs the result of every fragmentation operation, indicating whether the packets are fragmented into correct-sized fragments.

Caution This command should be used with extreme caution in a production environment, because of the amount of output that is created.

FRF.12 Frame Relay Fragmentation

This topic describes when FRF.12 can be used and how FRF.12 affects VoIP packets.

FRF.12 Frame Relay Fragmentation

FRF.12 specifies fragmentation of Frame Relay data frames:

- Frame Relay data frames that exceed the specified fragmentation size are fragmented.
- Smaller time-sensitive packets can be interleaved.
- This is the recommended Frame Relay fragmentation method to be used with VoIP over Frame Relay.
- Fragments VoIP over Frame Relay packets if the fragment size is set to a value smaller than the voice packet size.
- FRF.12 requires FRTS or DTS.

© 2009 Cisco Systems, Inc. All rights reserved.QoS v2.3--8-10

FRF.12 is the recommended fragmentation to be used with VoIP over Frame Relay while FRF.11 Annex C is the recommended fragmentation to be used with VoFR.

The FRF.12 Implementation Agreement defines FRF.12 fragmentation, and was developed to allow long data frames to be fragmented into smaller pieces and interleaved with real-time frames. In this way, real-time voice and non-real-time data frames can be carried together on lower-speed Frame Relay links without causing excessive delay and jitter to the real-time traffic like VoIP.

Because Frame Relay is a Layer 2 protocol, it has no way to tell which frame contains voice (VoIP) or data. Therefore, Frame Relay will fragment all packets larger than the fragment size into smaller frames, including VoIP packets. In a VoIP over Frame Relay network, it is important to configure the fragment size on the Data-Link Connection Identifier (DLCI) so that VoIP frames will not get fragmented. For example, a G.711 VoIP packet without cRTP is 200 bytes long. For this DLCI, do not set the fragment size to less than 200 bytes.

Frame Relay permanent virtual circuits (PVCs) not configured for VoFR use normal Frame Relay (FRF.3.1) data encapsulation. If fragmentation is turned on for this DLCI, it uses FRF.12 for the fragmentation headers.

FRF.12 specifies three types of fragmentation, of which only one form is supported in Cisco IOS Software:

- **End-to-end:** Supported in Cisco IOS Software
 - Packets contain the FRF.12 fragmentation header
 - Occurs at the PVC level
 - LMI packets are not fragmented
- **User-Network Interface (UNI):** Not supported in Cisco IOS Software releases
- **Network-to-Network Interface (NNI):** Not supported in Cisco IOS Software releases

FRF.12 is configured on a per-PVC basis. If one PVC or subinterface has fragmentation but another does not on the same physical interface, large packets can and will delay smaller packets. For applications where fragmentation is needed, you must turn on fragmentation on all PVCs or subinterfaces that carry large packets. FRF.12 fragmentation is configured within a Frame Relay map class using the **frame-relay fragment** command. The configured map class is then associated with the specific DLCIs. In addition, you must enable Frame Relay traffic shaping (FRTS) on the interface in order for FRF.12 fragmentation to work.

Cisco IOS Software Release 12.1(5)T introduces a distributed version of FRF.12 Frame Relay fragmentation to use with Distributed Traffic Shaping (DTS).

Configuring FRF.12 Frame Relay Fragmentation

This topic describes the Cisco IOS commands that are required to configure FRF.12.

Configuring FRF.12 Frame Relay Fragmentation

`router(config)#`

`map-class frame-relay map-class-name`

- Specifies a map class to define QoS values for a virtual circuit

`router(config-map-class)#`

`frame-relay fragment fragment-size`

- Enables fragmentation of Frame Relay frames for a Frame Relay map class
- Sets the maximum fragment size in bytes

`router(config-if)# | (config-subif)#`

`frame-relay class name`

- Associates a map class with an interface or subinterface

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3--8-11

FRF.12 fragmentation is configured within the Frame Relay map class. The **frame-relay fragment** command sets the maximum fragment size in bytes. On an interface, the **frame-relay class** command applies the map class to the interface or subinterface. To associate a map class with a DLCI, use the **class** command in Frame Relay DLCI configuration mode or Frame Relay VC-bundle-member configuration mode.

FRF.12 requires FRTS to be enabled.

Example: FRF.12 Frame Relay Fragmentation

FRF.12 Frame Relay Fragmentation Example

```
interface serial 0/0
  encapsulation frame-relay
  frame-relay traffic-shaping
!
interface serial 0/0.1 point-to-point
  frame-relay interface-dlci 100
  class FRF12
!
map-class frame-relay FRF12
  frame-relay fragment 80
!FRTS parameters
  frame-relay cir 64000
  frame-relay bc 2600
  frame-relay fair-queue
```

© 2009 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-12

The figure shows a configuration example where FRF.12 fragmentation is applied to a data Frame Relay circuit configured on the serial 0/0.1 subinterface.

The maximum fragment size is set to 80 bytes. This would be used in a VoIP over Frame Relay environment.

FRTS is enabled on the interface and the FRTS token bucket parameters are configured within the Frame Relay map class. In this figure, FRTS is enabled with a Committed Information Rate (CIR) of 64 kb/s, a committed burst (Bc) (normal burst size) of 640 bits, and uses weighted fair queuing (WFQ) as the shaping queue.

Note The purpose of fragmentation is to decrease serialization delays so that large data frames cannot block small voice packets for excessive periods of time. Both voice and data packets are subject to the shaping rate. FRTS token bucket operation is based on burst of credits in bits. If a large data packet gets through the shaper and uses up all the credits, the voice packets will be delayed until more credit becomes available. But if only a small fragment of a data packet can get through the shaper, the voice packet will be delayed for less time. So with FRTS and FRF.12, the bottleneck is at the shaper, not at the port. Therefore, the fragment size should be determined based on the CIR and not the physical interface clocking rate.

FRTS doesn't take into account the flags and cyclic redundancy checks (CRCs) in the shaping algorithm. Because a flag and CRC need to be added for every fragment, on lower-speed links (768 kb/s and below), you should shape to a value slightly lower than the actual CIR to account for the flags and CRCs. For example, on a 128-kb/s Frame Relay PVC, the recommended frame-relay CIR value to use will be 124,872 b/s with a Bc of 1250 bits per committed time window (Tc).

PVC Speed	Fragment Size (for 10-ms Delay)	Recommended CIR	Recommended Bc (CIR/100)
56 kb/s	70 bytes	52968 b/s	530 bits
64 kb/s	80 bytes	60952 b/s	610 bits
128 kb/s	160 bytes	124872 b/s	1250 bits
256 kb/s	320 bytes	252832 b/s	2530 bits
512 kb/s	640 bytes	508816 b/s	5090 bits
768 kb/s	960 bytes	764940 b/s	7560 bits

Monitoring FRF.12 Frame Relay Fragmentation

This topic describes the Cisco IOS commands that are required to monitor FRF.12.

Monitoring FRF.12 Frame Relay Fragmentation

```
router>
```

```
show frame-relay fragment [interface interface [DLCI]]
```

- Displays information about the Frame Relay fragmentation

```
router>show frame-relay fragment
interface    dlci  frag-type  frag-size  in-frag  out-frag  dropped-frag
Serial0/0.1  100   end-to-end  80         0        0         0
```

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3-8-13

The **show frame-relay fragment** command displays information about the FRF.12 Frame Relay fragmentation process. The fragment type will always display end-to-end because this is the only type currently supported on Cisco IOS Software. In addition to fragment type, the fragment size in bytes and associated DLCI is displayed.

Monitoring FRF.12 Frame Relay Fragmentation (Cont.)

```
router>
```

```
show frame-relay pvc [interface interface] [dlci]
```

- Displays statistics about PVCs for Frame Relay interfaces

```
router>show frame-relay pvc 100
```

```
PVC Statistics for interface Serial0/0 (Frame Relay DTE)
DLCI = 100, DLCI USAGE = LOCAL, PVC STATUS = INACTIVE, INTERFACE = Serial0/0.1
<output omitted>
Current fair queue configuration:
Discard      Dynamic      Reserved
threshold  queue count  queue count
    64         16         0
Output queue size 0/max total 600/drops 0
fragment type end-to-end fragment size 80
cir 64000      bc 2600      be 0      limit 325      interval 40
mincir 32000   byte increment 320  BECN response no  IF_CONG no
frags 0        bytes 0        frags delayed 0      bytes delayed 0

shaping inactive
traffic shaping drops 0
```

The **show frame-relay pvc** command output includes settings related to the FRF.12 fragmentation process. This output shows the fragment size (80 bytes in this example) used on the Frame Relay PVC. The fragment type is end-to-end because Cisco IOS Software currently supports only end-to-end FRF.12 fragmentation.

Summary

This topic summarizes the key points discussed in this lesson.

Summary

- Cisco LFI options are Multilink PPP with interleaving, FRF.12, and FRF.11 Annex C.
- Serialization delay is a function of the link speed and packet size. To ensure low delay and jitter for voice packets on slow links, the recommended standard goal for serialization delay is to not exceed 10 to 15 ms.
- The **ppp multilink** command enables MLP on a PPP interface. The **ppp multilink interleave** command enables interleaving of fragments within the multilink connection. The **ppp multilink fragment-delay** command specifies the maximum desired fragment delay for the interleaved multilink connection.
- The **show interface multilink** command output includes MLP LFI statistics information and indicates whether MLP interleaving is enabled on the interface.

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3—8-16

Summary (Cont.)

- FRF.12 specifies fragmentation of Frame Relay data frames when they exceed a specified size or when there are smaller time-sensitive packets that can be interleaved, and will fragment VoIP if fragment size is set to a smaller value than the VoIP size.
- The **frame-relay fragment map-class** command sets the maximum fragment size in bytes. On an interface, the **frame-relay class** command applies the map class to the interface, subinterface, or a DLCI.
- The **show frame-relay fragment** command displays information about the FRF.12 Frame Relay fragmentation statistics.

© 2003 Cisco Systems, Inc. All rights reserved.

QoS v2.3—8-16

Module Summary

This topic summarizes the key points discussed in this module.

Module Summary

- Link efficiency mechanisms are used to increase the perceived throughput and decrease the latency on WAN links.
- Class-based TCP and class-based RTP header compression enable header compression on specific traffic classes using the MQC method.
- LFI improves the serialization delay for packets leaving the router and prevents the voice packets from waiting on large data packets to be processed.

The various link efficiency mechanisms available in Cisco IOS Software include class-based TCP and RTP header compression, Layer 2 payload compression, MLP link fragmentation and interleaving (MLP LFI), and Frame Relay Fragmentation (FRF.12). You can use these QoS mechanisms on slower WAN links, such as 768 kb/s and slower, to improve the link efficiency by increasing the throughput and decreasing the overall delay.

Class-based TCP and RTP header compressions are configured using the MQC method to reduce the protocol header overhead. Header compression is most effective for packets with small payload size.

Layer 2 payload compression is not used much because of the software compression delay overhead. Using a hardware compression module can help reduce the software compression delay.

MLP LFI is supported on leased-line interfaces, while FRF.12 (for VoIP) and FRF.11 Annex C (for VoFR) are supported on Frame Relay interfaces. LFI is used to reduce the serialization delay of large packets. Serialization delay is based on the link speed and packet size. To ensure high voice quality, the recommended serialization delay is 10 to 15 ms maximum. As a general rule, a fragment size of 80 bytes for every 64 kb/s of link speed is recommended.

References

For additional information, refer to these resources:

- For more information on link efficiency mechanisms, refer to *Link Efficiency Mechanisms Overview* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/lem_oview_ps6350_TSD_Products_Configuration_Guide_Chapter.html
- For more information on Layer 2 payload compression, refer to *Understanding Data Compression* at this URL:
http://www.cisco.com/en/US/tech/tk713/tk802/technologies_tech_note09186a00801b3b86.shtml
- For more information on TCP and RTP header compression, refer to these RFCs:
 - RFC 2508 (RTP header compression)
 - RFC 1144 (TCP header compression)
 - RFC 2507 (IP header compression)
- For more information on FRF.12 and DTS configuration, refer to *Frame Relay Traffic Shaping With Distributed QoS on the Cisco 7500 Series* at this URL:
http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a96b5.shtml
- For information on FRF.11 Annex C, refer to *VoFR Encapsulation and Fragmentation* at this URL:
http://www.cisco.com/en/US/tech/tk1077/technologies_tech_note09186a00800942e1.shtml
- For more information on Cisco class-based TCP and RTP header compression, refer to *Configuring Class-Based RTP and TCP Header Compression* at this URL:
http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/config_cb_hdr_comp_ps6350_TSD_Products_Configuration_Guide_Chapter.html

Module Self-Check

Use the questions here to review what you learned in this module. The correct answers and solutions are found in the Module Self-Check Answer Key.

- Q1) With Layer 2 payload compression, what can be done to improve the compression/decompression delay of the router? (Source: Understanding Link Efficiency Mechanisms)
- A) enable CEF switching
 - B) enable fast switching
 - C) use the Stacker or Predictor compression algorithm
 - D) use hardware-assisted compression
- Q2) Which two of these statements are true, based on the following **show** output? (Choose two.) (Source: Configuring Class-Based Header Compression)
- ```
wg6r1#show policy-map interface s0/1
Serial0/1
Service-policy output: cust1
Class-map: voip (match-all)
 300 packets, 202035 bytes
 5 minute offered rate 2000 bps, drop rate 0 bps
Match: protocol rtp
Queuing
 Strict Priority
Output Queue: Conversation 264
Bandwidth 384 (kbps) Burst 9600 (Bytes)
 (pkts matched/bytes matched) 232/139975
 (total drops/bytes drops) 15/15795
compress:
header ip rtp
UDP/RTP compression:
Sent: 285 total, 284 compressed,
9086 bytes saved, 176014 bytes sent
1.5 efficiency improvement factor
99% hit ratio, five minute miss rate 0 misses/sec, 0 max
rate 2000 bps
```
- A) Both class-based TCP and RTP header compression are enabled for the VoIP traffic class.
  - B) IP payload compression is enabled for the VoIP traffic class.
  - C) LLQ is enabled for the VoIP traffic class.
  - D) Class-based RTP header compression is enabled for all RTP traffic.

- Q3) Which two of these statements are correct regarding TCP and RTP header compression? (Choose two.) (Source: Configuring Class-Based Header Compression)
- A) Hardware-assisted header compression is required to reduce the header compression delay.
  - B) TCP header compression compresses both the IP and TCP headers.
  - C) RTP header compression compresses the IP, UDP, and RTP headers.
  - D) TCP and RTP header compression can compress the respective protocol headers down to 10 bytes.
- Q4) TCP header compression is most effective on which of these applications? (Source: Configuring Class-Based Header Compression)
- A) VoIP
  - B) FTP
  - C) video streaming
  - D) Telnet
  - E) TFTP
- Q5) RTP header compression is most effective on which of the following applications? (Source: Configuring Class-Based Header Compression)
- A) FTP
  - B) VoIP
  - C) TFTP
  - D) Telnet
  - E) HTTP
- Q6) Which three statements about FRF.12 and FRF.11 Annex C are correct? (Choose three.) (Source: Configuring Link Fragmentation and Interleaving)
- A) FRF.12 fragmentation is used on DLCIs configured for VoIP.
  - B) FRF.12 fragmentation is used on DLCIs configured for VoFR.
  - C) FRF.11 Annex C fragmentation is used on DLCIs configured for VoIP.
  - D) FRF.11 Annex C fragmentation is used on DLCIs configured for VoFR.
  - E) FRF.12 fragmentation requires FRTS to be enabled.
  - F) Cisco IOS Software supports UNI and NNI FRF.12 fragmentation, and not end-to-end fragmentation.
- Q7) When configuring Multilink PPP with interleaving, where is the fragment size configured? (Source: Configuring Link Fragmentation and Interleaving)
- A) The fragment size in milliseconds is configured under the physical serial interfaces with PPP encapsulation and Multilink PPP enabled.
  - B) The fragment size in milliseconds is configured under the logical multilink interface.
  - C) The fragment size in bytes is configured under the map class.
  - D) The fragment size in bytes is configured within the policy map.
- Q8) Which two factors will influence the serialization delay? (Choose two.) (Source: Configuring Link Fragmentation and Interleaving)
- A) link speed
  - B) speed of light in the media
  - C) router CPU processing power
  - D) packet size

- Q9) To ensure good voice quality, what is the recommended fragment size? (Source: Configuring Link Fragmentation and Interleaving)
- A) 80 bytes per every 64 kb/s of the clocking, which will result in a 10-ms serialization delay
  - B) 40 bytes per every 64 kb/s of the clocking, which will result in a 20-ms serialization delay
  - C) 20 bytes per every 64 kb/s of the clocking, which will result in a 10-ms serialization delay
  - D) 120 bytes per every 64 kb/s of the clocking, which will result in a 150-ms serialization delay
- Q10) From the **show interface multilink** output, which state(s) should be in the open state to indicate proper Multilink PPP operation over an IP interface? (Source: Configuring Link Fragmentation and Interleaving)
- A) LCP, Multilink, and IPCP
  - B) LCP, IPCP, LFICP
  - C) LCP and MLP
  - D) LCP and IPCP

## Module Self-Check Answer Key

|      |         |
|------|---------|
| Q1)  | D       |
| Q2)  | C, D    |
| Q3)  | B, C    |
| Q4)  | D       |
| Q5)  | B       |
| Q6)  | A, D, E |
| Q7)  | B       |
| Q8)  | A, D    |
| Q9)  | A       |
| Q10) | A       |