

QoS

CISCO

QoS

(642-642)



**Smarter
Training**

This LearnSmart exam manual covers the most important topics you will encounter on the Quality of Service exam (QoS - 642-642). By studying this manual, you will become familiar with an array of exam-related topics, including:

- IP QoS Fundamentals
- IP QoS Components
- Congestion Management Methods
- QoS Best Practices
- And more!

Give yourself the competitive edge necessary to further your career as an IT professional and purchase this exam manual today!

Cisco CCVP - Quality of Service

LearnSmart Exam Manual

Copyright © 2011 by PrepLogic, LLC
Product ID: 011480
Production Date: July 19, 2011

All rights reserved. No part of this document shall be stored in a retrieval system or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein.

Warning and Disclaimer

Every effort has been made to make this document as complete and as accurate as possible, but no warranty or fitness is implied. The publisher and authors assume no responsibility for errors or omissions. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this document.

LearnSmart Cloud Classroom, LearnSmart Video Training, Printables, Lecture Series, Quiz Me Series, Awdeco, PrepLogic and other PrepLogic logos are trademarks or registered trademarks of PrepLogic, LLC. All other trademarks not owned by PrepLogic that appear in the software or on the Web Site (s) are the property of their respective owners.

Volume, Corporate, and Educational Sales

Favorable discounts are offered on all products when ordered in quantity. For more information, please contact us directly:

1-800-418-6789
solutions@learnsmartsystems.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Abstract	8
Your Product	8
About the Author.....	8
Domain 1 - IP QoS Fundamentals	9
QoS Basics	9
<i>The Need for QoS</i>	9
Traffic Characteristics	12
<i>Bandwidth</i>	12
<i>Delay</i>	12
<i>Jitter</i>	14
<i>Loss</i>	14
QoS Implementation	15
<i>QoS Implementation Methods</i>	15
Domain 2 - IP QoS Components.....	20
QoS Models	20
<i>Best-Effort</i>	20
<i>Integrated Services</i>	20
<i>Differentiated Services</i>	20
DiffServ Functionality.....	21
<i>Classification and Marking</i>	21
<i>Congestion Management and Avoidance</i>	26
<i>Link Efficiency</i>	27
<i>Traffic Policing and Shaping</i>	27
<i>Per-Hop Behaviors (PHB)</i>	27
<i>Trust Boundaries</i>	29
Domain 3 - Modular QoS CLI (MQC) and Auto-QoS	29
MQC CLI Basics	29
MQC Examples	29
<i>Voice Only</i>	29
<i>Voice Using Access List and class-default</i>	31
<i>Matching Opposites</i>	32
<i>Matching multiple criteria</i>	33
AutoQoS Basics.....	34

AutoQoS Implementation.....	34
AutoQoS (VoIP) and AutoQoS (Enterprise) Differences	36
Configuration	37
<i>policy-map</i>	37
<i>class-map</i>	37
<i>class</i>	37
<i>service-policy input</i>	37
<i>service-policy output</i>	37
<i>match protocol</i>	38
<i>match fr-dlci</i>	38
<i>match access group</i>	38
<i>match cos</i>	38
<i>match precedence</i>	38
<i>match dscp</i>	39
<i>match input-interface</i>	39
<i>match ip rtp</i>	39
<i>match mpls experimental</i>	39
<i>match mpls experimental topmost</i>	39
<i>match packet length</i>	40
<i>set atm-clp</i>	40
<i>set cos</i>	40
<i>set precedence</i>	40
<i>set dscp</i>	40
<i>set fr-de</i>	41
<i>set ip tos</i>	41
<i>set mpls experimental imposition</i>	41
<i>set mpls experimental topmost</i>	41
<i>bandwidth (CBWFQ)</i>	41
<i>random-detect</i>	42
<i>random-detect dscp</i>	42
<i>random-detect precedence</i>	43
<i>random-detect exponential-weighting-constant</i>	43
<i>compression header ip</i>	43
<i>auto discovery qos</i>	44

<i>auto qos</i>	44
<i>auto qos voip</i>	44
<i>auto qos voip cisco-phone</i>	44
Troubleshooting	45
<i>show class-map</i>	45
<i>show policy-map</i>	45
<i>show ip tcp header-compression</i>	46
<i>show ip rtp header-compression</i>	46
<i>show auto discovery qos</i>	47
<i>show auto qos</i>	48
<i>show policy-map interface</i>	48
Domain 4 - Classification and Marking	49
Classification and Marking Basics	49
Classification and Marking Configuration	50
<i>Network Based Application Recognition (NBAR)</i>	50
<i>NBAR Configuration</i>	51
<i>Basic Classification and Marking Configuration</i>	52
Pre-Classify	56
Trust Boundaries	56
Configuration	56
<i>ip nbar protocol-discovery</i>	56
<i>ip nbar pdlm</i>	56
<i>ip nbar custom</i>	57
<i>qos pre-classify</i>	57
Domain 5 - Congestion Management Methods	58
Congestion Management Mechanisms	58
<i>First In, First Out (FIFO) Queuing</i>	58
<i>Priority Queuing (PQ)</i>	59
<i>Weighted Round Robin (WRR) and Custom Queuing (CQ)</i>	60
<i>Weighted Fair Queuing (WFQ)</i>	61
<i>Class Based – Weighted Fair Queuing (CBWFQ)</i>	63
<i>Low Latency Queuing (LLQ)</i>	64
Control Plane Policing (CoPP)	65
Software and Hardware Queuing	65

Congestion Management Configuration	65
<i>fair-queue</i>	65
<i>hold-queue</i>	66
<i>bandwidth (CBWFQ)</i>	66
<i>max-reserved-bandwidth</i>	66
<i>priority</i>	66
<i>queue-list queue limit</i>	67
<i>queue-list queue byte-count</i>	67
<i>queue-list protocol</i>	67
<i>queue-list interface</i>	68
<i>queue-list default</i>	68
Domain 6 - Congestion Avoidance Methods	68
Tail-Drop	68
Weighted Random Early Detection (WRED)	68
<i>WRED and Queuing</i>	70
<i>Explicit Congestion Notification (ECN) and WRED</i>	73
Configuration Avoidance Configuration	74
<i>random-detect</i>	74
<i>random-detect dscp</i>	74
<i>random-detect precedence</i>	75
<i>random-detect ecn</i>	75
<i>random-detect exponential-weighting-constant</i>	75
Domain 7 - Traffic Policing and Shaping	76
Token Bucket Concept	76
Traffic Policing Basics	76
Traffic Policing Configuration	77
<i>Basics</i>	77
<i>police</i>	78
Traffic Shaping	81
<i>Basics</i>	81
<i>Shaping Adaptation</i>	82
Traffic Shaping Configuration	82
<i>shape</i>	83
<i>shape percent</i>	83

<i>shape fecn-adapt</i>	83
<i>shape fr-voice-adapt</i>	83
Troubleshooting	84
Domain 8 - Link Efficiency Mechanisms	84
Link and Fragmentation and Interleaving (LFI)	84
<i>Multilink PPP (MLP)</i>	84
<i>Frame Relay LFI (FRF.12)</i>	85
<i>LFI Configuration</i>	85
Compression	87
<i>Header Compression</i>	87
<i>Payload Compression</i>	87
<i>Compression Configuration</i>	88
Configuration	88
<i>ppp multilink</i>	88
<i>ppp multilink fragment delay</i>	88
<i>ppp multilink interleave</i>	89
<i>ppp multilink group</i>	89
<i>frame-relay traffic-shaping</i>	89
<i>frame-relay class</i>	89
<i>frame-relay fragment</i>	89
<i>frame-relay traffic rate</i>	90
<i>frame-relay adaptive-shaping</i>	90
Domain 9 - QoS Best Practices	90
QoS Application Type Requirements	90
<i>Voice</i>	90
<i>Video</i>	91
<i>Data</i>	91
Campus QoS Recommendations	92
Customer Edge (CE) to Providers Edge (PE) QoS Recommendations	92

Abstract

The Cisco Certified Voice Professional is one of the most well respected certifications in the world. By attaining it, students and candidates signify themselves as extremely accomplished and capable Networking and Voice over IP Professionals. The five exams required to become a CCVP are extremely difficult and not to be taken lightly. They cover a myriad of topics, in particular - the QoS(Quality of Service) exam covers the process of managing the quality of network flow cQoSrol and packet consolidation. The QoS is multiple choice, simulative, and incorporates test strategies such as "drag and drop" and "hot area" questions to verify a candidate's knowledge.

Before taking this exam, you should be very familiar with both Cisco technology and networking. You must have also attained the Cisco CCNA certification by passing either the one or two part path.

Your Product

This QoS Exam Manual has been designed from the ground up with you, the student, in mind. It is lean, strong, and specifically targeted toward the candidate. Unlike many other QOS products, the LearnSmart QOS Exam Manual does not waste time with excessive explanations. Instead, it is packed full of valuable techniques, priceless information, and brief, but precisely worded, explanations. While we do not recommend using only this product to pass the exam, but rather a combination of LearnSmart Audio Training, Practice Exams, and Video Training, we have designed the product so that it and it alone can be used to pass the exam.

About the Author

Sean Wilkins is an accomplished networking consultant and has been in the field of IT since the mid 1990's working with companies like Cisco, Lucent, Verizon and AT&T. In addition to being a CCNP and CCDP, Sean is also a MCSE and an overall "IT expert". In addition to working as a consultant, Sean spends a lot of his time as a technical writer and editor.

Domain 1 - IP QoS Fundamentals

QoS Basics

The important thing about Quality of Service (QoS) is its purpose. Networks these days are getting bigger and bigger and a larger amount of companies are using the Internet in some way to transport their traffic. Many companies have grown to the point where the differentiation of traffic over their networks is vital in order for certain traffic to get where it is going in a timely manner. Within QoS there are a number of mechanism types which are used to remedy these problems. Within networks there are two main ways to mark traffic. Marking the priority of the traffic is important because it provides some information which is part of the traffic itself, which enables the networking equipment to look for the traffic, and allows the networking equipment to perform prioritizing actions on the important traffic. The second way is a way of notating congestion in a network. This type of marking is mainly used for congestion avoidance because the routers are given some indication of when congestion starts and, from this information, have the option to use preempt congestion by dropping packets based on a congestion threshold.

The routers themselves can be configured to make QoS decisions based on the information in the traffic but there is typically no end-to-end QoS mechanism. Cisco routers use a number of different mechanisms to provide QoS. These mechanisms include congestion management solutions, queuing mechanisms, congestion avoidance mechanisms, and traffic control mechanisms.

Bottom line is that QoS is the grouping of technologies which provides a way to specify different quality parameters to different types of traffic. QoS is needed in today's networks because more and more networks are becoming data and voice converged networks, meaning that data, voice and video flow over the same network. This of course makes data, voice and video traffic influential over each other. Some traffic, like file transfers or web pages, is not as reliant on specific timing as VoIP or Video over IP. If a packet of a file arrives 30 seconds late, the transfer is just slowed down a little. If a voice or video packet arrives late it becomes useless in the audio or video stream.

The Need for QoS

In order to deal with QoS issues, there have been a number of different types of mechanisms created to help implement QoS. These different mechanisms work in different ways in order to be used in conjunction with each other so end-to-end QoS is under the requirements needed for the application.

Video and Voice

With a voice call, one-way delay budgets have been established by standards. These budgets set a total amount of delay which is considered acceptable for different levels of voice service. These are shown in Table 1:

1-way Delay (in ms)	Description
0-150	ITU G.114 Acceptable Range
0-200	Cisco Acceptable Range
150-400	ITU G.114 Degraded Range
400+	ITU G.114 Unacceptable Range

Table 1 - Delay Budget

On top of the requirements for delay, a minimal amount of sustained bandwidth is needed in order for a voice or video call to go through and be of good quality. The amount of bandwidth needed and the quality needed depend on the codec selection. The following are the main voice codecs which are used today:

Voice Codec	Acronym	Name	Bit Rate
G.711	PCM	Pulse Code Modulation	64-kbps
G.722	SB-ADPCM	Sub-Band ADPCM	48, 56, 64-kbps
G.722.1	MLT	Modulated Lapped Transform	24 and 32-kbps
G.722.2	ACELP	Algebraic Code Excited Linear Prediction Coder	6.60, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85-kbps
G.723.1 (5.3-kbps)	ACELP	Algebraic Code Excited Linear Prediction Coder	5.3-kbps
G.723.1 (6.3-kbps)	MP-MLQ	Multi Pulse-Maximum Likelihood Quantization	6.3-kbps
G.726	ADPCM	Adaptive Differential Pulse Code Modulation	16, 24 and 32-kbps
G.728	LDCELP	Low Delay Code Excited Linear Prediction	16-kbps
G.729	CS-ACELP	Conjugate Structure Algebraic CELP	8-kbps
G.729A	CS-ACELP Annex A	Conjugate Structure Algebraic CELP Annex A	8-kbps

Table 2 - ITU Voice Codecs

Video is also being used more and more, and the quality of this depends on the same factors as with voice, the amount of delay and the amount of bandwidth. The following shows a list of the most commonly used video codecs today:

Video Codec	Name	Bit Rate
MPEG-1	Moving Picture Experts Group, Version 1	500 to 1500 kbps
MPEG-2	Moving Picture Experts Group, Version 2	1.5 to 10 Mbps
MPEG-4	Moving Picture Experts Group, Version 4	28.8 to 400 kbps
H.261	Video Coding Experts Group	100 to 400 kbps

Table 3 - Common Video Codecs

In order for the demands of these different types of traffic to be sustained, some amount of QoS must be implemented over the network to ensure that the requirements are met.

Both voice and video can also be split into two main categories, interactive and non-interactive. Interactive traffic is communication which goes in two directions in response to each other. Interactive traffic requirements are stricter because delay, jitter and delay requirements must be low to keep the conversation ongoing and flowing. This is not true of non-interactive traffic like audio or video presentations since these are typically in one direction; the delay, loss and delay are not noticed.

TCP Windowing

There is a congestion mechanism within TCP which is used to try to limit congestion on the network. The main controlling option within this mechanism is the use of *windowing*. Windowing works by allowing a TCP connection to send a specific amount of traffic without having to receive an acknowledgement. By default, the window size is set to 536 bytes. What this means is that 536 bytes of traffic is sent and an acknowledgement is expected from the destination. If an acknowledgement is received then the window size is increased by 2 times. If this traffic is sent and an acknowledgement is received, then the window size is increased by 3 times, and so on until the maximum window size is reached. If, at any point, an acknowledgement is not received, the window size is cut in half.

If a QoS mechanism is not put in place to cut down the chance of packet loss then the window size will stay low and make TCP connections very inefficient. The mechanism which would be used for this on Cisco equipment is Weighted Random Early Detection (WRED).

Traffic Characteristics

Bandwidth

Bandwidth is simply the amount of data that can be sent over a network at one time. Bandwidth is the easiest part of QoS to understand; to use more bandwidth than is available on the network, some or all of the traffic will be affected. When thinking of bandwidth in QoS terms, it is typically available bandwidth that needs to be considered. Available bandwidth is a measurement of the minimum bandwidth available on a path from point A to point B, divided by the number of potential traffic flows. This is shown in the following figure:



Figure 1 - Bandwidth Example

Using this figure, the amount of minimum bandwidth is 10-Mbps across the whole path. If ten flows are needed, the total available bandwidth per flow is 1-Mbps.

On Cisco equipment there are two different concepts which must be clearly defined. There are two commands, **bandwidth** and **clock-rate**, on Cisco equipment. The **bandwidth** command is used to set the amount of bandwidth that the equipment will use for calculations. These include dynamic routing protocols, load and statistics. By default, the bandwidth is set to 1.544 Mbps (or a T1 Rate). The **clock-rate** command is used to set the actual line rate of the interface. A good example of this is a frame relay link. It is possible to have a physical access rate of 1.544 Mbps with a bandwidth of 512 kbps. The 512 kbps from the provider equates to a Committed Information Rate (CIR) of 512 kbps.

Delay

Delay is a crucial part of QoS management. The amount of overall delay from end-to-end is very important when dealing with voice and video over networks. There are many different things that can affect the amount of delay that is introduced from one side of a path to the other. The nine main delay factors are: processing delay, queuing delay, serialization delay, propagation delay, shaping delay, network delay, codec delay, compression delay and jitter buffer delay.

Processing or forwarding delay is the amount of time it takes the layer 3 devices (router or switch) to transfer a packet in one interface and out another. Many different things affect this, including CPU speed, CPU utilization, total memory, available memory, and bus speed, among others.

Queuing delay is the amount of time a packet spends in the queue of a layer 3 device. Queues are used in equipment to store data when the bandwidth is currently completely utilized. This information is stored for a short time in a queue until the bandwidth opens up. The amount of time that the packet spends in these queues is the queuing delay.

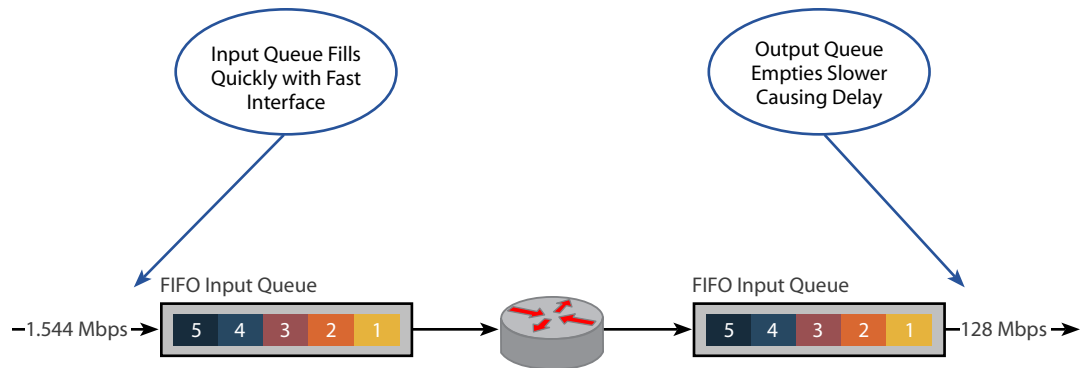


Figure 2 - Queuing Delay Example

Serialization delay is the amount of time it takes for a packet to be broken down into layer 2 frames, then into layer 1 electric or optical signals. The amount of time taken for a packet to be broken down for a specific link can be calculated with the formula from Figure 3.

$$\frac{\text{\# of bits sent}}{\text{Link Speed}}$$

Figure 3 - Serialization Delay Formula

Propagation delay is the amount of time it takes for a packet to cross the physical medium. The amount of time that is taken for a packet to be transmitted over media can be roughly calculated with the formula in Figure 4.

$$\frac{\text{Length of Link (meters)}}{2.1 \times 10^8 \text{ meters per second}}$$

Figure 4 - Speed of Light over Copper/Fiber

Shaping delay is the delay that is introduced when a piece of equipment shapes the flow of traffic. As an example, think of what happens when traffic comes into a fast interface and has to go out a slow interface. Without shaping, there is a high potential that some of this traffic will be queuing delayed then tail dropped, meaning that the traffic would fill the queue and then overflow and drop future packets. In order to try to stop this, traffic shaping can be configured to average out the traffic flow. In this case, traffic shaping would try to queue the faster traffic and slow down the flow until incoming traffic slows. This slowing in the traffic introduces shaping delay. Assuming adequate queue depth, this is quite useful because it tries to allow the largest amount of packets to reach their destination.

Network delay is the delay that is introduced by the cloud. The cloud is typically a part of the network which is not under direct control, so trying to find the exact delay can be challenging. Typically this number is given as a maximum delay that is given as part of the network contract.

There are two things which typically cause *codec delay*, traffic packetization and *look ahead*. Traffic packetization is the process of taking an analog voice or video signal and translating that into a digital signal. The *look ahead* feature is specific to the codecs which use this feature (i.e. G.729). This feature is used with predictive codecs which delay the stream enough to receive part of the next voice packet.

Compression delay is the amount of time it takes to compress a packet which introduces a small amount of delay to traffic. However, the compression itself can also decrease other types of compression, like serialization and others. Serialization delay goes down because there is a smaller overall packet, thus less to breakdown and send.

The last type of delay is *de-jitter buffer delay* or *initial playout delay*. The first question should be, what is jitter? Jitter is simply the variation of packet delay. What this means is that some packets come in quickly and some come in more slowly causing havoc for the receiver trying to put the traffic back together. In order to mitigate this, Cisco equipment implements a jitter buffer which is used to smooth out the traffic coming in by storing it for a short time in a buffer and emptying at an average pace. The short amount of time that the traffic spends in this queue causes additional delay. By default, Cisco gateways are setup with a 40 ms read ahead delay.

Jitter

Jitter or delay variation is when the amount of delay changes from packet to packet, which causes packets to arrive at the destination out of order as determined by the RTP time-stamping. Obviously, when dealing with a voice or video call, the packets must be reassembled in the correct order or the voice or video would not make any sense. Some devices have what is called a jitter buffer which is used to mitigate small amounts of jitter by essentially creating a small queue of packets which can be used to reorganize the packets into order. This can only be done correctly however when the overall amount of jitter is minimal. Overall, queuing and network delay tend to be the worst jitter offenders. A queuing mechanism which reduces the time it takes for a voice or video packet to be forwarded should be chosen to reduce jitter.

Loss

Loss is simple. It is the complete loss of a packet somewhere across the path from A to B. There are a number of different reasons for loss which include; output drop, input drop, overruns, ignored frames, and frame errors, among others.

Output drop or *tail drop* is when a router is trying to transfer a packet from an input queue to an output queue after routing and finding the output queue to be completely full. If this happens, the packet is dropped.

Input drop is when a router tries to receive a packet but its input queue is completely full.

Overruns happens when a router is trying to receive a packet but the router is so busy that it is unable to allocate buffer space for the packet. This does not mean that the buffer is full, simply that the CPU did not have the time to create it.

Ignored frames are frames which are dropped because there is no layer 2 buffer space available to put them.

Frame errors occur when a frame is received but the CRC or checksum do not match, which shows that the frame was corrupted in some way along the link.

QoS Implementation

When implementing QoS, three main steps are used: traffic identification, traffic classification, and traffic policy definition. Traffic identification is the beginning of the process where the network designers must determine the different types of traffic using the network. With this information, the network designers will create traffic parameters which are needed for each class of traffic. In the traffic classification step, the network designers create the traffic classes and setup a way to differentiate each type of traffic (TCP/UDP port number, IP address, ...). In the traffic policy definition step, the network designer must create policies for each type of traffic, which includes bandwidth and delay requirements. This can also include different queuing mechanisms which are used for each type of traffic. Congestion detection and avoidance mechanisms can be used within the policy to try to minimize the effects from congestion on the network.

QoS Implementation Methods

CLI

Command-Line Interface (CLI) commands individually are always an option with Cisco IOS. The CLI implementation method is the older way of configuring QoS on a per-interface basis. It would, of course, be quite taxing to replicate code from router to router and from interface to interface.

MQC

Modular QoS Command-Line Interface (MQC) is the newer way of configuring QoS at the CLI level. With MQC, the traffic is separately put in classes, the policies are separately configured, and then the policies are applied ingress or egress at the interface level. The three main commands to complete these tasks are **class-map**, **policy-map**, and **service-policy**.

AutoQoS

AutoQoS is a newer feature of Cisco IOS. It enables the router to monitor the traffic being processed across the router's interfaces and automatically setup traffic classes, policies, and enables them per interface. AutoQoS is covered in more detail in the AutoQoS domain.

SDM QoS Manager

Cisco's Security Device Manager (SDM) is a newer interface which has been created to ease the amount of knowledge required to administrator Cisco routers. This is done through a web interface which can be installed on the router or on the PC. SDM provides an easy way of configuring QoS. The following figures show the configuration process through SDM:

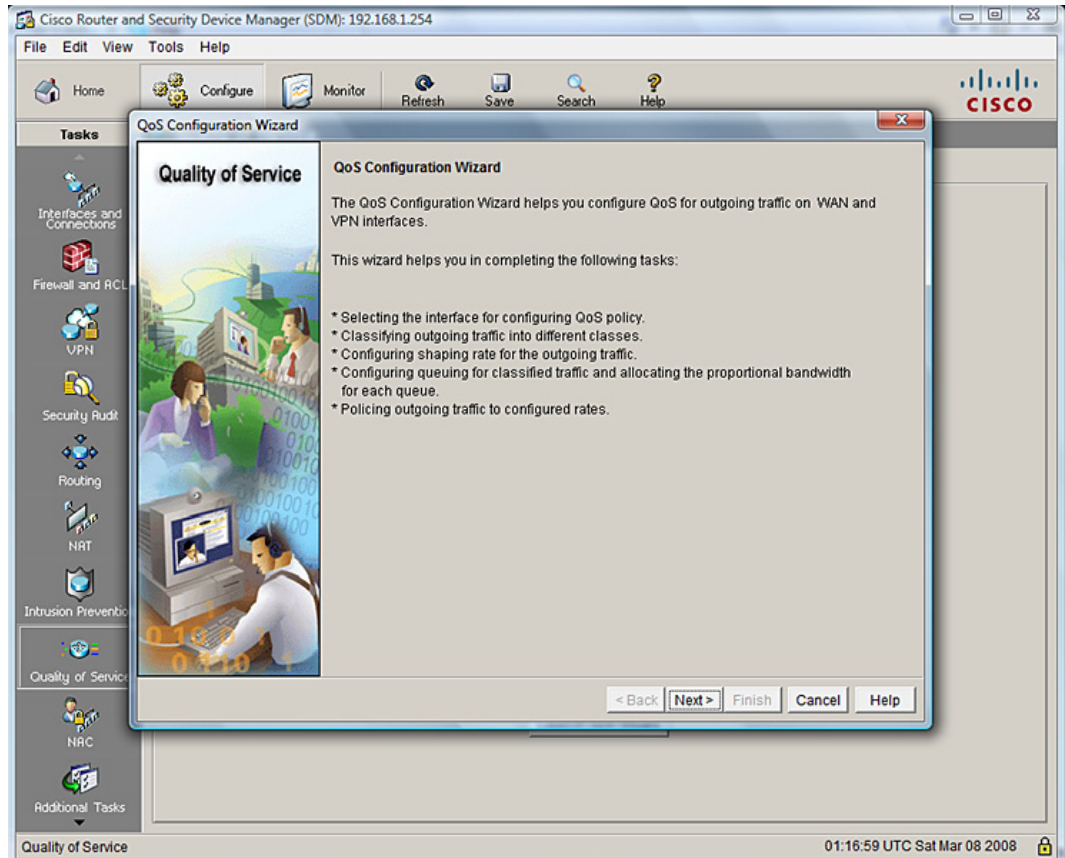


Figure 5 - Cisco SDM QoS Configuration Wizard

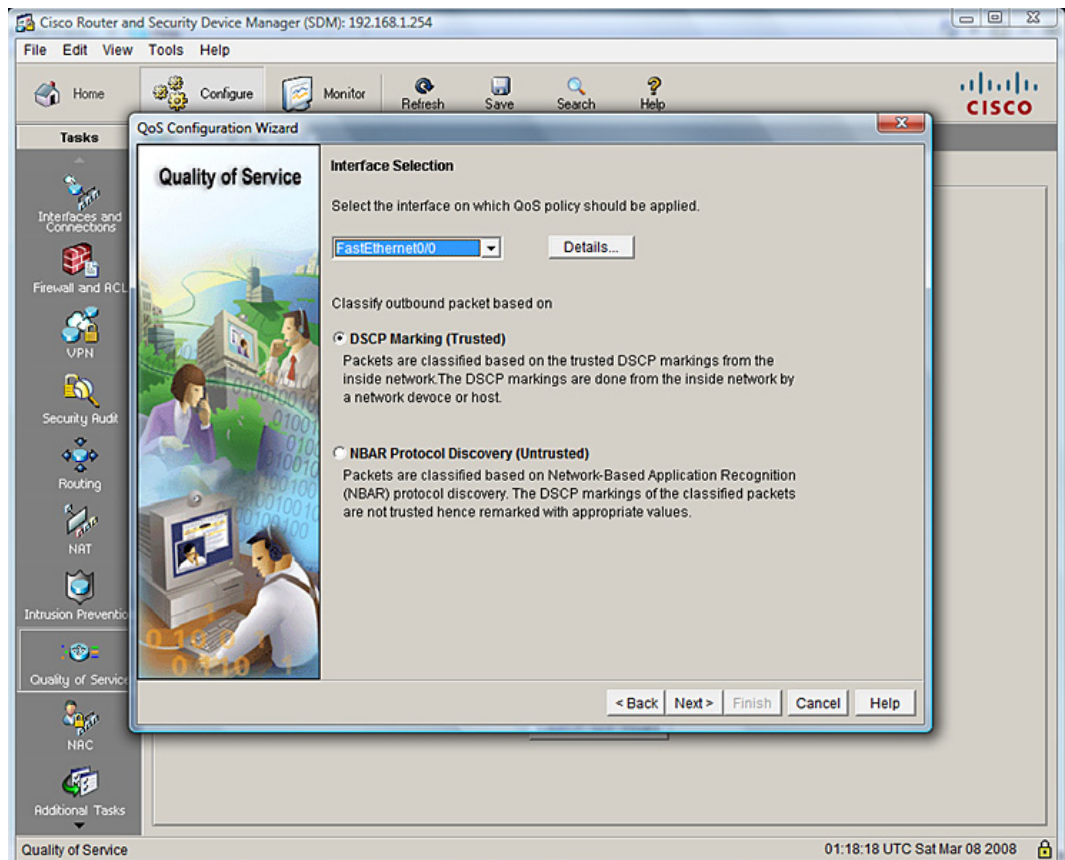


Figure 6 - SDM QoS Interface Configuration

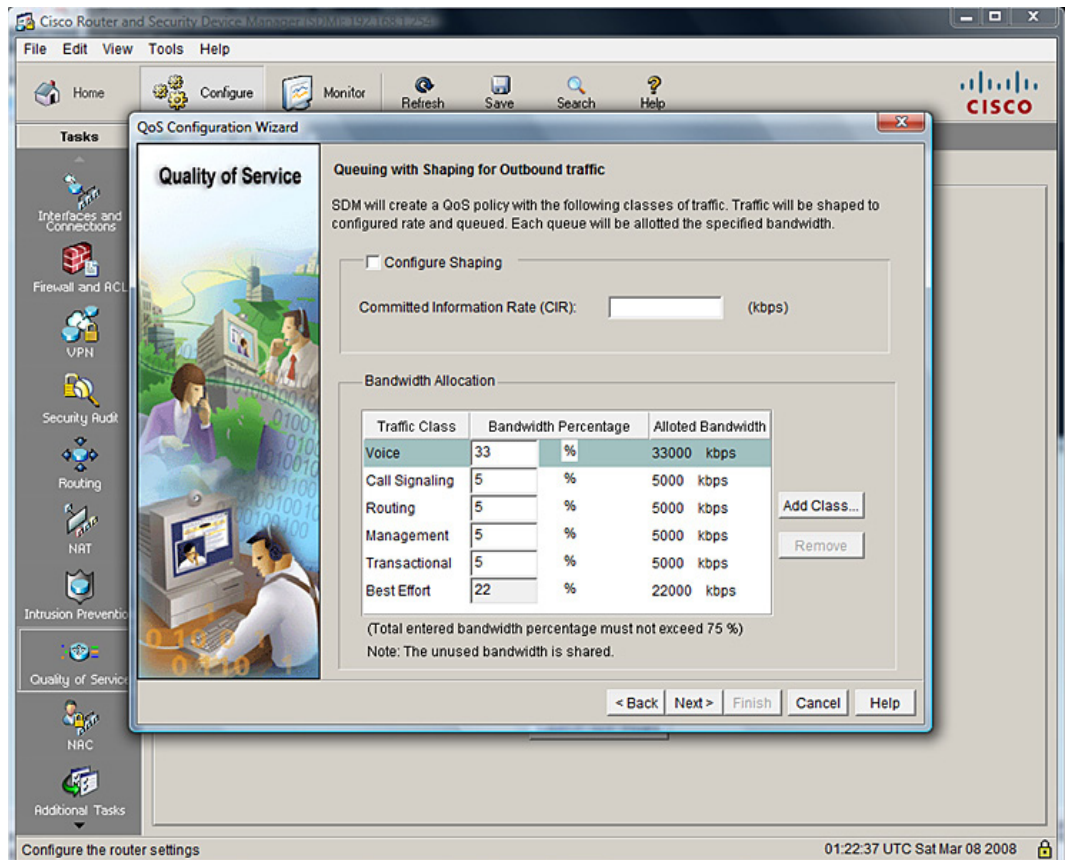


Figure 7 - SDM QoS Queuing and Shaping

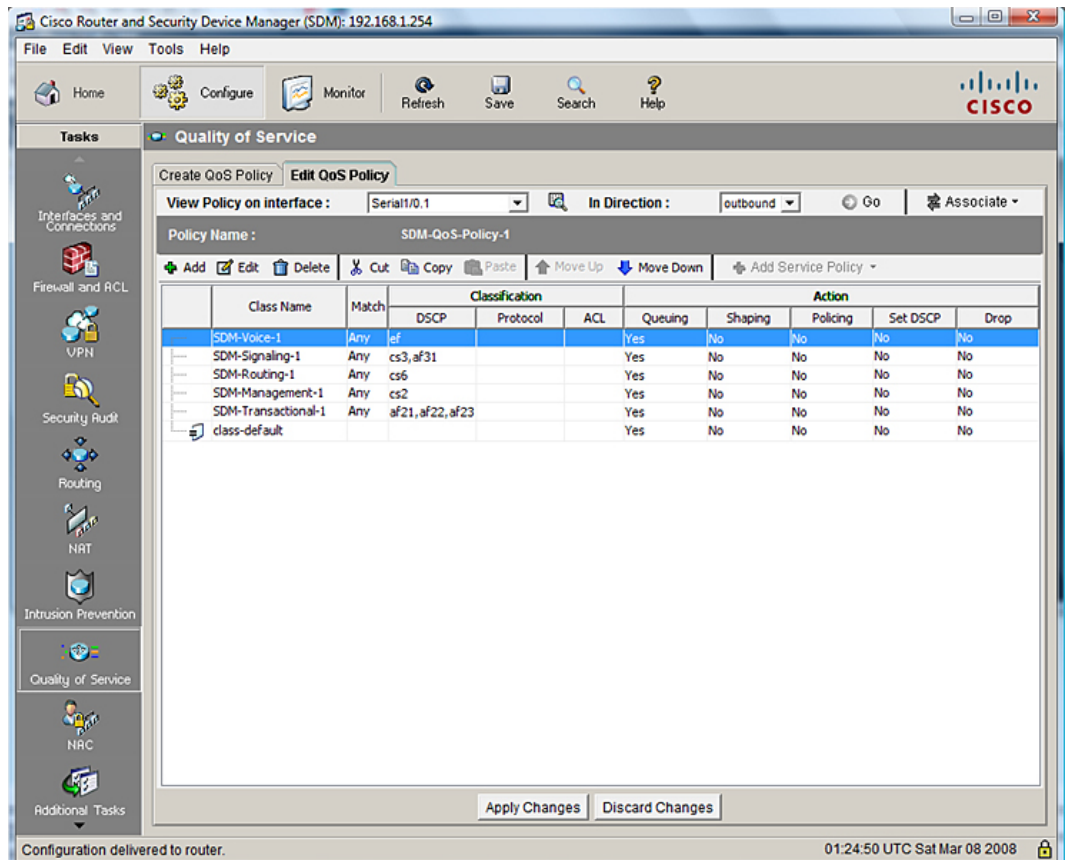


Figure 8 - SDM QoS Edit Policy

Domain 2 - IP QoS Components

QoS Models

A QoS model is a group of end-to-end capabilities that is given to certain types of traffic. Within Cisco IOS, there are three models which are supported: the Best-Effort model, the Integrated Services model, and the Differentiated Services model. Each has their specific advantages and disadvantages. The network designer is responsible for choosing a model which best fits the types of traffic to be run over the network. The following sections include a little detail on each of these models.

Best-Effort

This is the simplest model. Best-Effort means that data is processed on a first-come first-served basis. No traffic in this model gets preference. This is best used when the traffic on the network is limited to low priority file transfers, web traffic, as well as email traffic. Within Cisco IOS, this model is implemented through FIFO (First In, First Out) Queuing.

Integrated Services

The Integrated Service model is used when the end devices are aware of their resource requirements and have the ability to request specific traffic profiles before any data is sent. The network does not have any traffic profile setup initially until the end device requests it. Once the end device requests it, the network tries to comply by checking current availability. If the network is able to meet the requirements of the requested traffic profile, a confirmation is sent back to the end device. Once this confirmation is received, the end device starts to send traffic. The network will however keep track of the traffic. In the case where the end device exceeds the requested resources then the network has the option to drop over resource traffic. On Cisco IOS, this model can be implemented with Resource Reservation Protocol (RSVP). When using RSVP, the queue type is typically weighted fair queuing (WFQ) when specific bandwidths need to be maintained. In situations where the requirements are low delay and high throughput, Weighted Random Early Detection (WRED) can also be used.

Differentiated Services

The differentiated services model is a little different; there is no request for a specific traffic profile from the end device. The network is configured to give certain QoS parameters to matched traffic and the match type is configured on the networking equipment. Bandwidth management with differentiated service is typically done through policing mechanisms, which limit the amount of resources dedicated to each flow of traffic and traffic shaping, which shapes the traffic flow into an average bandwidth. On Cisco IOS, this can be done through Committed Access Rate (CAR) and Traffic Shaping configuration options.

Typically with DiffServ, traffic is marked on ingress to the network, so that the traffic can be correctly treated throughout as much of the network as possible. DiffServ requires the marking of the IP header which is at Layer 3 and is propagated throughout the network. DiffServ allows the traffic to be marked or remarked at different network boundaries, typically between an ISP and a customer. DiffServ also introduces the Differentiated Services Codepoint (DSCP) which reorganizes the ToS field in the IP header, allowing up to 64 different classifications. These classifications are defined as Per Hop Behaviors (PHB) which are defined and explained later in this domain.

DiffServ Functionality

Classification and Marking

The first step in configuring DiffServ is to classify the traffic. This is the step where different priorities of traffic are put into classes which will give them the QoS parameters to work best through the network. Classification can be done based on a number of different criteria, including IP address, TCP/UDP port numbers, incoming interface (ingress), as well as more parameters like source and destination, application and a number of other factors which can be identified based on deep packet inspection. On Cisco equipment this can be done either through the use of access lists (the old way) or through the use of the **class-map** command (MQC). The **class-map** command has the ability to utilize the **match** command; this enables the use of access-lists, different traffic descriptors, and the use of NBAR for classification. The **class-map** and **match** commands also give the ability to classify based on multiple conditions.

The second step uses the information found in the classification step and marks the traffic so that equipment throughout the network can make decisions based on the markings. DiffServ uses a number of different **set** commands which are able to change any number of the markings detailed in the next sections.

Class of Service (CoS)

Ethernet provides prioritization options, but an added header must be used because the original Ethernet frame does not include a field for prioritization (See Figur). In order to provide for prioritization, a priority field was added to the 802.1Q frame. This field is defined by 802.1P and it is 3-bits long (See Figure 10). These 3-bits provide for eight different levels of class which can be defined for each Ethernet frame.

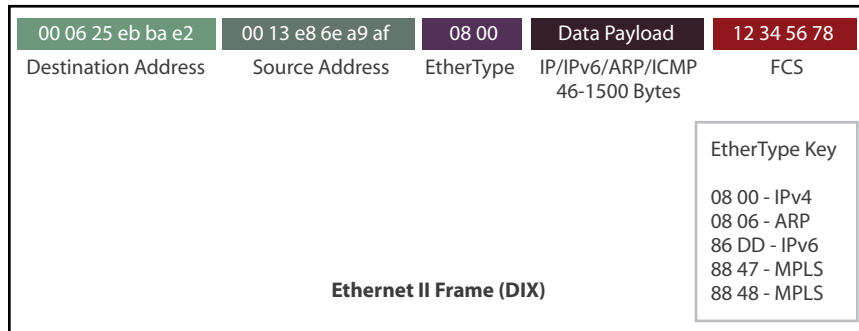


Figure 9 - Ethernet II Frame

With the 802.1Q/P frame, the EtherType of the frame is changed to Hex 8100 and the original EtherType is included inside the 802.1Q/P Tag. This can be seen in Figure 22.

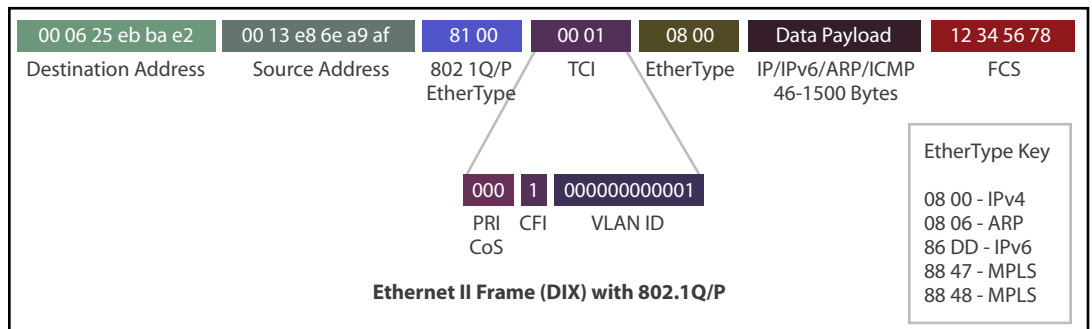


Figure 10 - IEEE 802.1Q/P Frame

Figure 11 shows how the different Classes are typically used. Cisco IP phones automatically send out 802.1Q/P frames with a CoS of 5. This is then used by the Layer 2 equipment to prioritize it over typical data which, if marked, is typically marked with a CoS of 0.

CoS (Bits)	CoS (Decimal)	RFC791	Application
000	0	Routine	Best-Effort
001	1	Priority	Medium Priority Data
010	2	Immediate	High Priority Data
011	3	Flash	Call Signaling
100	4	Flash-Override	Video Conference
101	5	Critical	Voice
110	6	Internetwork	Reserved
111	7	Network Control	Reserved

Figure 11 - IEEE 802.1P CoS (PRI) Options1

Frame Relay - Discard Eligible (DE)

Within Frame Relay there are a couple of mechanisms which are used to denote both priority and congestion. These include the Forward Explicit Congestion Notification (FECN), Backward Explicit Congestion Notification (BECN) and the Discard Eligible fields. The FECN and BECN fields are used to notify the different frame relay equipment that congestion has occurred during the traffic's transmission. FECN is used when congestion occurs outbound and is marked towards the destination. BECN is used when the destination equipment receives a frame with the FECN bit set to 1. The networking equipment then sets the BECN bit to 1 for all traffic sent back towards the origin. The DE bit is used to denote a frame with lower priority. If the DE bit is set to 0 then the frame has *normal* priority. If it is set to 1 then the frame is considered of *lower* priority. The layout of these bits can be seen in figure 12.

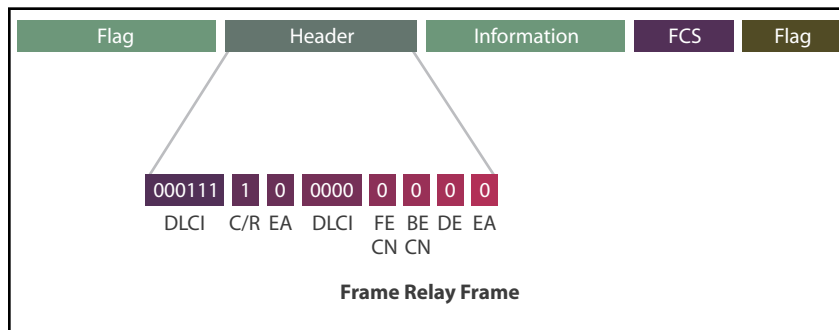


Figure 12 - Frame Relay Header

ATM - Cell Loss Priority (CLP)

Within Asynchronous Transfer Mode (ATM), there is a bit which is used similarly to the Frame Relay DE bit; this is called Congestion Loss Priority (CLP) bit. The CLP works similarly to the DE bit. If it is set, the cell is considered of lower priority. This bit is typically set by the ATM switch when congestion on specific traffic goes above a configured threshold. Within ATM, there are two different main types of cell headers, one for the User-to-Network (UNI) and one for Network-to-Network (NNI). These are shown in figure 13.

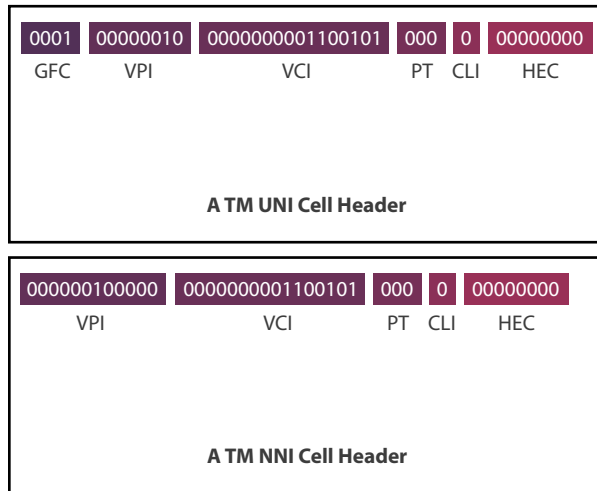


Figure 13 - ATM Cell Headers (UNI & NNI)

MPLS EXP

Unlike the previous sections, MPLS is considered a protocol which provides functionality of both Layer 2 and Layer 3 protocols. This is why it is typically called a Layer 2.5 protocol. MPLS can be implemented in a number of ways with a number of different technologies. In Figure 14, MPLS is shown running with an Ethernet header. MPLS works by applying a label or tag to a frame/cell that provides for the features that MPLS provides, mainly VPN and traffic engineering. MPLS provides a field called the Experimental field (Exp) which is used to provide CoS. Originally this field was intended to parallel the values of the Type of Service (ToS) – IP Precedence field, which includes 3-bits to notate the class of the traffic. However, currently this field in the IP header is used in a different way. Instead of using a 3-bit field for IP Precedence, the new standard provides a 6-bit field to notate Class. This field is called the Differentiated Service Codepoint (DSCP). Because of this new standard, the network architects must consider how to use the MPLS Exp field at ingress.

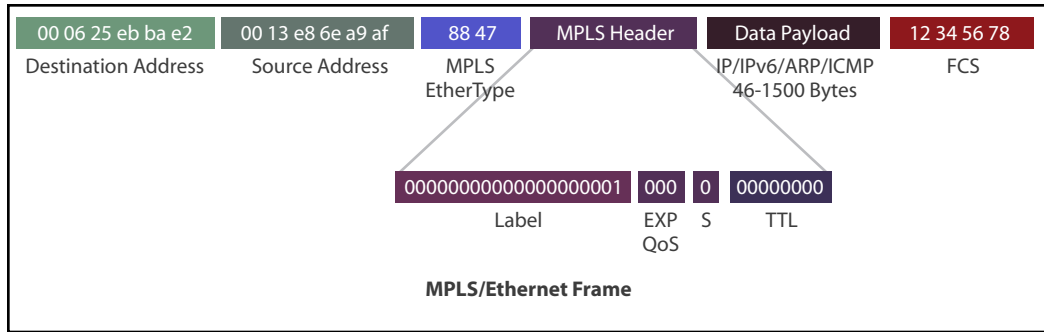


Figure 14 - MPLS Label (Ethernet Frame)

ToS – IP Precedence

At Layer 3, there is a field which is defined within IP which allows QoS marking. This field within the IP header is called the Type of Service (ToS) field.

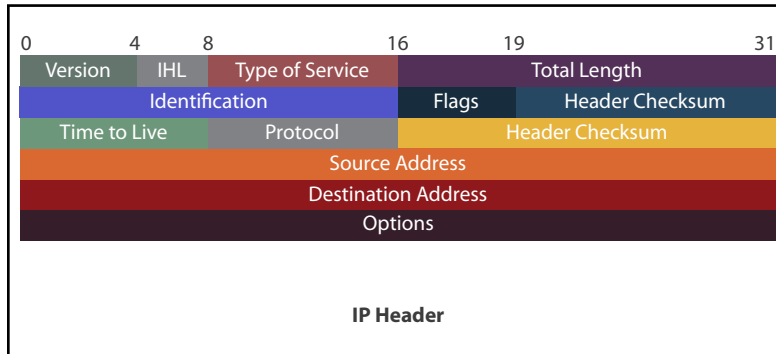


Figure 15 - IP Header

As stated in the primer, there are two main ways this field can be used. The original way used the field by splitting it into an IP Precedence – 3-bits, Delay – 1-bit, Throughput – 1-bit, Reliability – 1-bit, and two bits reserved. The second way is described in the next section. The IP precedence bits were used to define quality based on eight different bit sequences. These are shown in Figure 16.

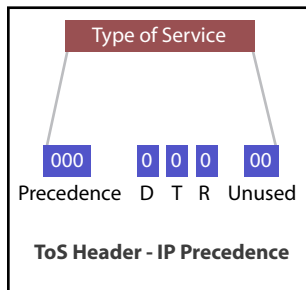


Figure 16 - IP Precedence

Bits	Definition
000	Routine
001	Priority
010	Immediate
011	Flash
100	Flash Override
101	Critical
110	Internetwork Control
111	Network Control

Figure 17 - IP ToS Precedence

The Delay bit was set to 1 when the QoS request needed *low delay*. The Throughput bit was set to 1 when the QoS request needed *high throughput*. The Reliability bit was set to 1 when the QoS request needed *high reliability*.

ToS - Differentiated Service Codepoint (DSCP)

As noted in the previous section, there is a second way to use the ToS field in the IP header. This is for the Differentiated Service Codepoint (DSCP), which is the current use of this field in most networks. DiffServ splits the ToS between a 6-bit DSCP field and a 2-bit Explicit Congestion Notification (ECN) field.

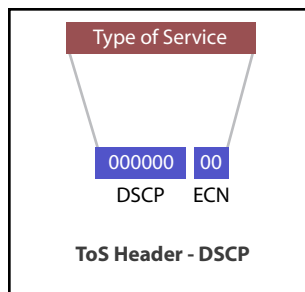


Figure 18 - DSCP

Congestion Management and Avoidance

Once the classification and marking steps have been accomplished at the edge of the network, there needs to be a mechanism which uses this information and puts it into practice. DiffServ does this by using congestion management and avoidance mechanisms. These mechanisms look for the marking information in the traffic processed and make decisions based on this information.

Queuing

Congestion management is implemented within DiffServ with queue management. There are a number of queue management mechanisms which can be implemented. Once implemented, these mechanisms control the way traffic is collected and forwarded through the networking equipment. Because congestion can cause the buildup of traffic on the networking equipment, the management of the queues which contain this buildup of traffic can greatly effect how this traffic is handled and processed through the equipment. There are a number of different queuing mechanisms which can be used to manage traffic. Each mechanism provides the management of traffic but implements it in different ways. The queuing mechanism which is used is based on the types and characteristics of traffic on the network.

By default, most interfaces on Cisco equipment run the First In, First Out (FIFO) queuing mechanism. This mechanism effectively does not give priority to any traffic. In order to give different types of traffic priority, a number of queuing mechanisms have been developed. Currently the mechanisms which are supported on most Cisco equipment are: FIFO, Priority Queuing (PQ), Custom Queuing (CQ), Weighted Fair Queuing (WFQ), Class Based – Weighted Fair Queuing (CBWFQ) and Low Latency Queuing (LLQ). These different mechanisms allow the network designer to prioritize the traffic in a number of ways depending on the requirements of the specific network. The specific queuing mechanisms available are discussed in detail in Domain 5.

Tail Drop

While not a congestion avoidance mechanism, *tail drop* is the default behavior for queues on Cisco equipment. *Tail drop* works by simply dropping the packets which are destined for an already full queue. The main problem with this is that no priority is given to any packet dropped, so high priority traffic which should not be dropped is, and low priority traffic which should be dropped is not.

Weighted Random Early Detection (WRED)

WRED is used as a replacement for tail drop. WRED provides a congestion avoidance mechanism. WRED does this by selectively dropping packets early before congestion occurs. This is done by tracking the average queue size and choosing to drop packets when the average queue depth exceeds a given threshold.

Link Efficiency

Multilink PPP (MLP)

MLP provides the capability to fragment and interleave packets together, allowing for time-sensitive packets to be forwarded out of the hardware queue (FIFO) faster, when large datagram packets can potentially be slowing these time-sensitive packets. It does this by splitting larger packets into fragments, which can then be interleaved with these time-sensitive packets.

Header Compression

Header compression on Cisco equipment has two forms: TCP header compression and RTP header compression. Both provide for compression which can afford significant savings in terms of bandwidth required. Both TCP and RTP header compression are usually implemented per interface but can also be implemented using the MQC policy, class and service maps. The major disadvantage of using header compression is that it adds delay to the connection. With TCP compression this may not be problem, but with traffic that uses RTP compression, this can become a big issue should the end-to-end delay budget be already high.

Payload Compression

Payload compression is another option which can be used to save traffic bandwidth. Cisco supports three different payload bandwidth options: Stacker, Predictor, and Microsoft Point-to-Point Compression (MPPC). Stacker compression is quite CPU intensive but does not require a large amount of memory and is supported in some Cisco hardware. Stacker also works with almost any layer 2 point-to-point encapsulation. Predictor compression is not as CPU intensive but requires higher memory intensity. Predictor only works on PPP and LAPB encapsulations. MPPC is only supported on PPP encapsulations and is used only between a Cisco device and a Microsoft client.

Traffic Policing and Shaping

Traffic policing is used to limit the amount of traffic that comes in on an interface. Traffic policing can be setup in a number of ways through the configuration of conform, violate and exceed policies. Traffic shaping is used to smooth out the traffic flow, essentially averaging the peaks and valleys of the traffic into a smoother line. Traffic shaping is used to try to prevent the loss of packets because of traffic exceeding the queue depth of the equipment or through the violation of policing policies.

Per-Hop Behaviors (PHB)

The DSCP field is used to specify a Per-Hop Behavior (PHB). A PHB specifies the forwarding treatment of the traffic. There are a number of PHBs and all specify different levels of service.

The first and most used PHB is the Default PHB which is specified when the DSCP is equal to '000000'. The Default PHB provides a best effort forwarding behavior which is the equivalent to no QoS. What this means is that all other traffic specified with other PHBs will get preference.

The second type of PHB is the Class Selector (CS) PHBs which are designated by 'xxx000', meaning that the last three (low order) binary digits must be '000'. The Class Selector PHBs are typically used to maintain backward compatibility with IP Precedence. The first three (high order) binary digits are allowed to be different values because the first three digits are used for IP Precedence. Class Selector PHBs or code-points are given a name based on the IP Precedence that they equate to: CS1 = IP Precedence 1 = '001000' through CS7 = IP Precedence 7 = '111000'. As a side note, the Default PHB is also considered a Class Selector PHB ('000000').

The third type of PHB is the Assured Forwarding (AF) PHB which provides a way to classify traffic into four classes AF1, AF2, AF3 and AF4. These classes equate to '001xxx', '010xxx', '011xxx' and '100xxx' in binary. The x's in the binary for AF PHBs are used to further classify the different classes of traffic into drop probability groups. These are specified as low drop ('xxx010'), medium drop ('xxx100') and high drop ('xxx110'). It must however be clear that the behavior of these classes is completely up to the configuration. There is no inherent priority between classes. In other words, by default, class AF1 does not have priority over AF2. Priority is configured through the assignment of queue space and bandwidth. Each class is configured with a specified amount of queue space and a percentage of the bandwidth that it is allowed to use. In order to prevent Tail Drop, WRED is used on the queues. It is also important to note that if the amount of bandwidth is not policed (traffic policing), it is possible for any class to use more bandwidth than configured for.

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

Table 4 - DSCP AF Values

The final type of PHB is Expedited Forwarding (EF) which provides for a traffic priority that promises low delay, loss and jitter, as well as a guarantee of bandwidth allotted. The EF PHB is specified as DSCP 46 or '101110'. It should be noted that as with AF PHB classes, the amount of bandwidth must be policed in order for the EF traffic to not monopolize the interface bandwidth.

The ECN field is used to notify the networking equipment that the traffic has or has not experienced congestion.

'00'	Non-ECN compatible equipment
'01' and '10'	ECN compatible equipment – No congestion
'11'	ECN compatible equipment – Congestion experienced

Table 5 - ECN Values

Trust Boundaries

Trust boundaries are setup within a network that uses QoS; specifically a trust boundary is used to set the point in the network where the markings of incoming traffic are trusted and not re-classified. This can be setup at any point in the network, but it is recommended that the trust boundary point be as close to the end system as possible. Normal end systems are typically not trusted with the exception of IP phones, but the access switch or access router is a good place to start should the equipment have the ability and processor power to classify and mark the traffic in accordance with the network policies.

Domain 3 - Modular QoS CLI (MQC) and Auto-QoS

- Explain how AutoQoS is used to implement QoS policy

MQC CLI Basics

The first thing that needs to be covered in this section is the purpose and functionality of the MQC. The MQC was developed to make configuration of both policy-based routing and QoS easier for the engineer. With the older Command Line Interface (CLI), traffic had to be specified and matched with access-lists. This has been found to be quite a bit of work if complicated matching is required. On top of these difficulties, CLI configuration did not have a method for dynamically classifying traffic through any type of deep packet inspection. MQC provides these features through a simpler configuration interface and through the use of technologies like Network Based Application Recognition (NBAR).

MQC started by defining three main phases of configuration. First the traffic must be classified; this is done through the use of the **class-map** command. The **class-map** configuration is done using the **match** commands. Multiple **match** commands can be used in order to correctly classify the different types of traffic on the network. The second phase is to create a traffic policy; this is done through the use of the **policy-map** command. Within the policy map configuration the class maps are assigned via the **class** command. The traffic policy allows multiple things to be configured. These include setting congestion or marking indicators, enabling policing or shaping of the traffic, enabling different types of traffic queues, enabling WRED, or just specifying the dropping of the traffic matched. All of these settings use various versions of the **set** command. The final part of the configuration that must be done is to attach the policy to a specific interface and traffic direction. This is done with the **service-policy** command. All of these commands are shown in detail in the configuration section of this domain.

MQC Examples

When configuring QoS, there are a number of things that must be known first. The main one is, what traffic is currently on the network? and which of this traffic needs to be separated for priority? Once this is complete, the configuration is ready to be started. The following sections will show examples of a couple of different common configurations.

Voice Only

One of the most common uses for QoS is to separate the voice and the data traffic on a network. There are a number of ways to configure using MQC depending on what is being searched for in the traffic. For Example, the traffic can be matched based on the use of Real-time Transport Protocol (RTP) through the use of the **match ip rtp** command. Voice signaling traffic can also be matched based on port numbers or through the use of NBAR. NBAR enables the networking equipment to recognize different protocols based on pre-programmed matching criteria. In order to utilize this functionality, use the **match protocol** command. The following shows the command to configure one class to be used for voice traffic based on RTP and classifying all other traffic into a default class.

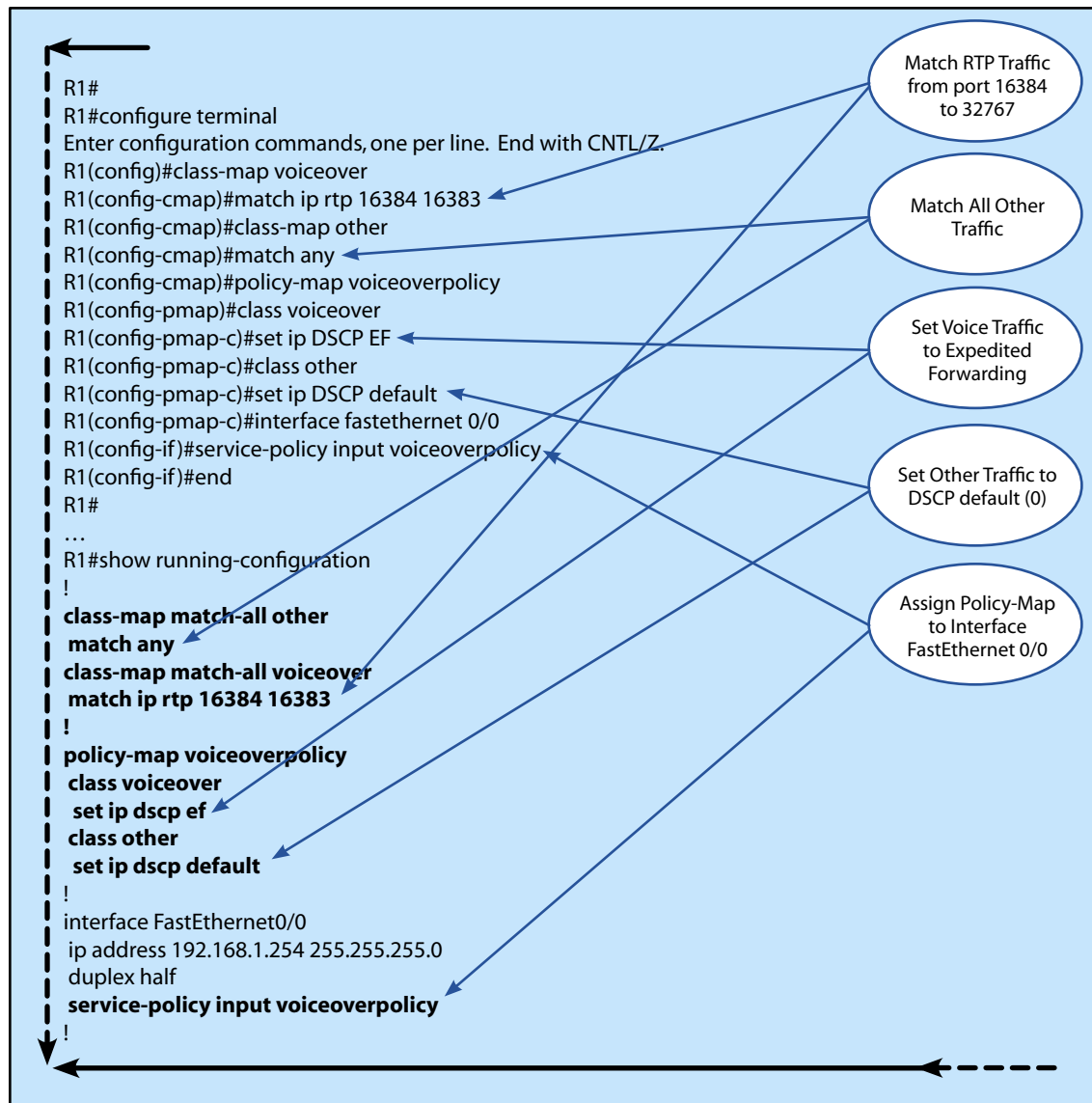


Figure 19 - MQC Example 1

Voice Using Access List and class-default

MQC also has the capability to utilize some of the advantages of access-lists. In order to use them the **match access-group** command is used. MQC also has the capability to support a default class; within MQC this class is called **class-default**. The following configuration example shows how these two options can be used.

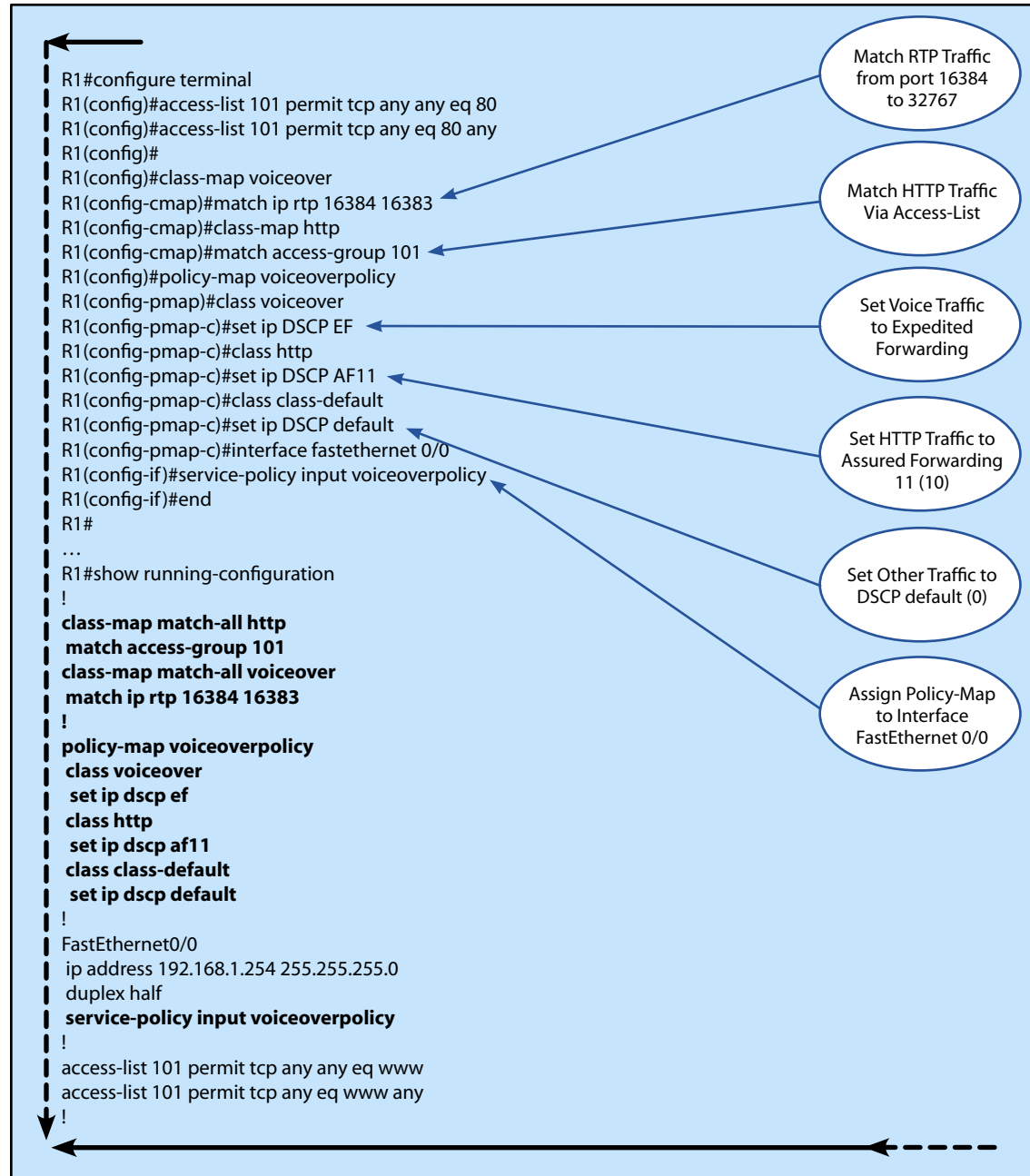


Figure 20 - MQC Example 2

Matching Opposites

Along with the **match** command, there is also a command that does the opposite, that is, not match to the access list. This is done through the use of the **match not** command. The following configuration example shows how the **match not** command is used.

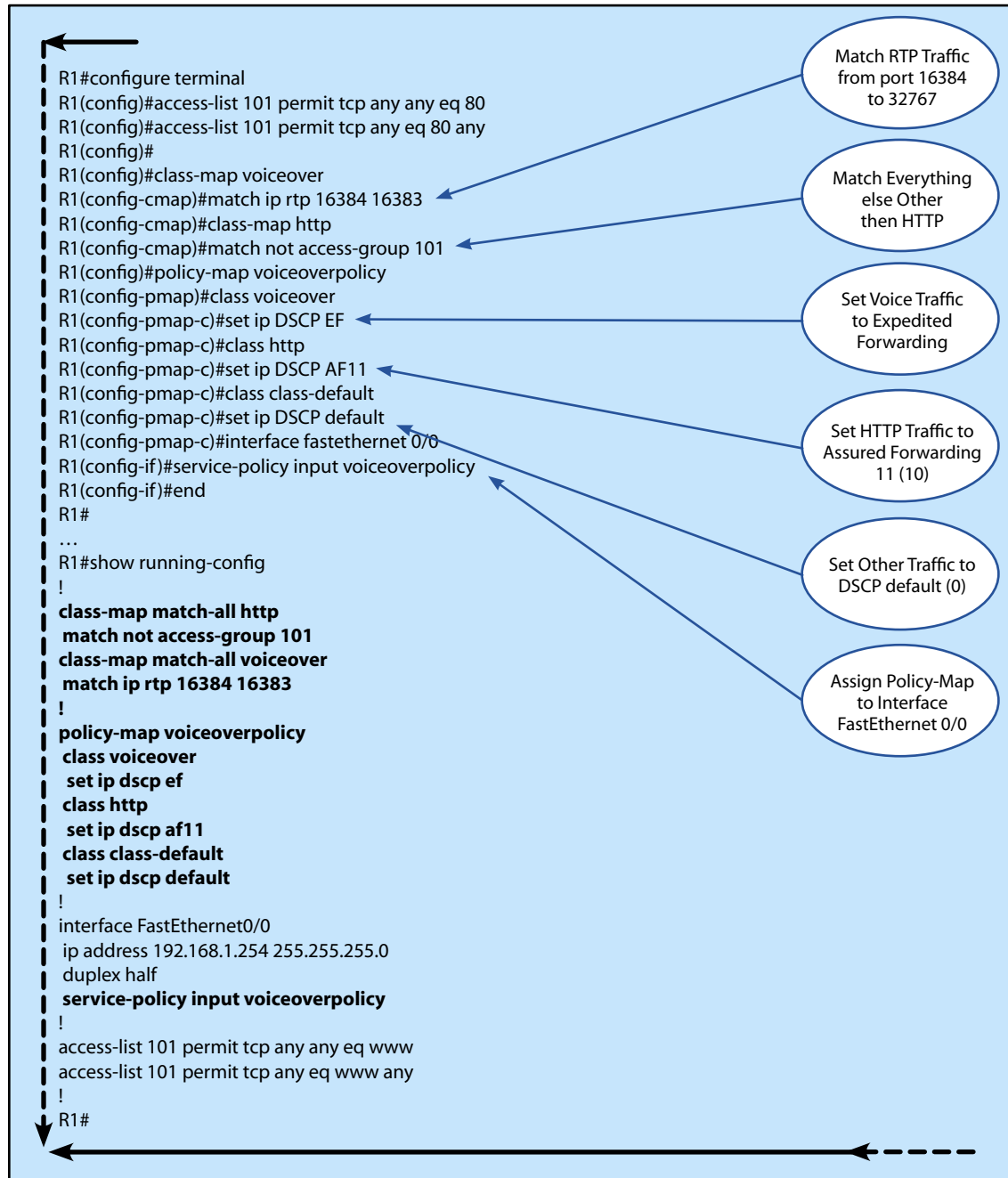


Figure 21 - MQC Example 3

Matching multiple criteria

Within MQC there are two main ways that criteria can be matched. Within each class map, the criteria to be matched can be matched when any of the criteria are matched, or matched when all of the criteria are matched. This means that if more than one **match** command is listed under a class map, there are two ways they can be interpreted. Either, any of the **match** commands designate a match, or all of the **match** commands designate a match. The following configuration example shows how these can be used.

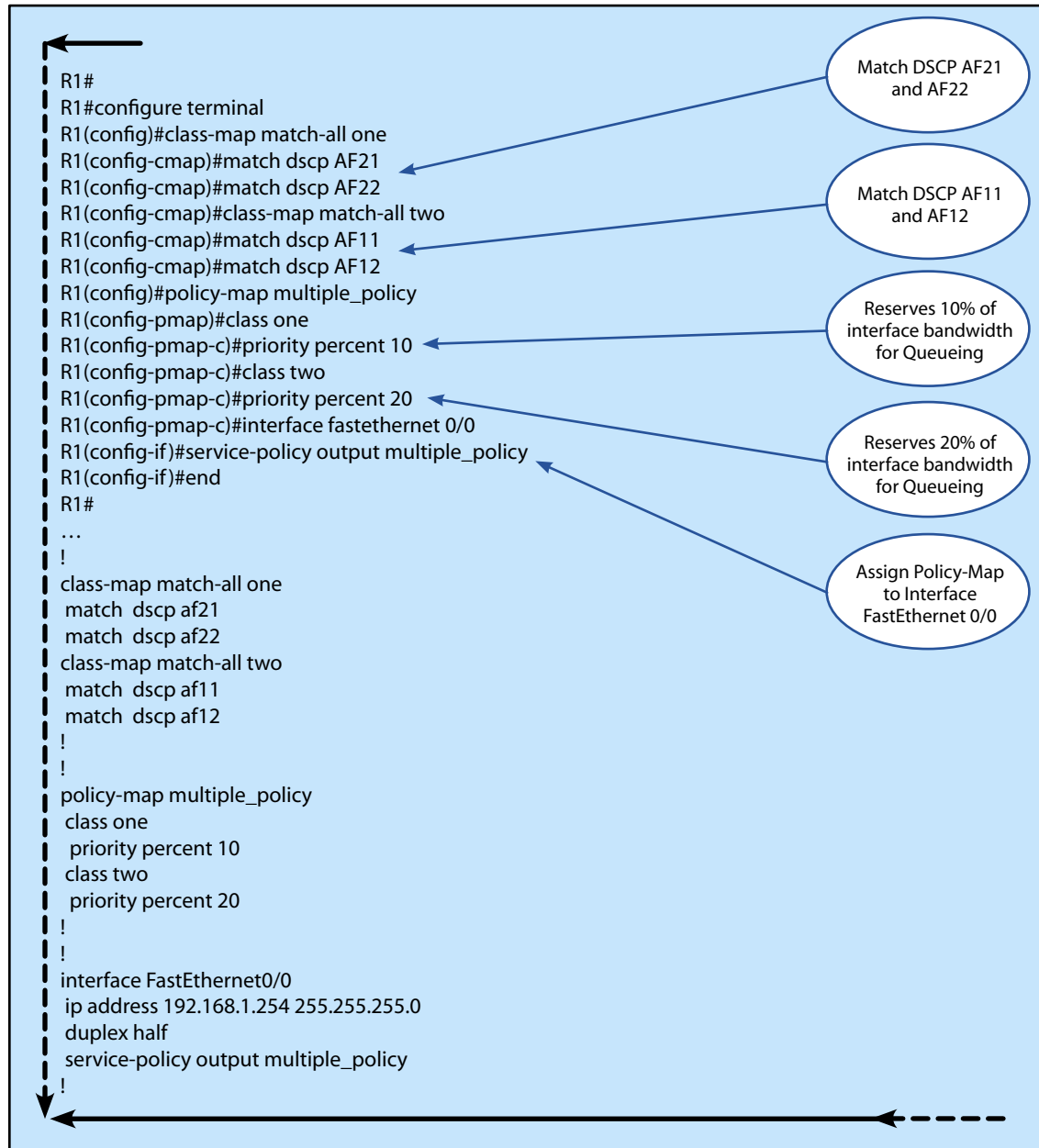


Figure 22 - MQC Example 4

AutoQoS Basics

AutoQoS is a newer feature of Cisco IOS. It enables the router to monitor the traffic being processed across the router's interfaces and automatically setup traffic classes, policies and enables them per interface. AutoQoS does this by monitoring the traffic on the router and watching for specific behavior. Once it has enough information it will automatically create the IOS configuration that suits the need of the traffic. AutoQoS has two different versions. The first, which was the initial release, is meant to only prepare the router for VoIP functions (Auto QoS VoIP) and the second is meant to discover many types of traffic and create QoS configurations. This second type is called AutoQoS in the Enterprise. AutoQoS has minimum requirements for its use: Cisco Express Forwarding (CEF) must be enabled, Network Based Application Recognition (NBAR) must be enabled, and correct bandwidths and IP addresses must be configured on the router's interfaces. In order for AutoQoS to work with Simple Network Management Protocol (SNMP), it must already be setup and the community "AutoQoS" must have write permission. To enable AutoQoS, the **auto qos voip** command should be configured on the interfaces. For AutoQoS in the Enterprise, the **auto discovery qos** and **auto qos** commands should be configured on the interfaces.

AutoQoS is recommended for use on small to mid-sized networks where the in-depth expertise that needed for QoS configuration may be lacking. This is because of how AutoQoS works. AutoQoS generates a configuration based on the types of traffic that are currently being monitored on the network. This monitoring and recognition is done via NBAR. The configuration is then generated in a command line format that mirrors the MQC type configuration. This generated configuration is then able to be tweaked by the network engineer should it need it.

AutoQoS in the Enterprise takes AutoQoS to another level in that NBAR protocol discovery is used to determine the types of traffic going through the interfaces of a router. AutoQoS in the Enterprise also has two distinct stages. The first stage involves the protocol discovery and traffic monitoring and the second stage involves the actual implementation of the policies (should it be deemed the discovery correct). NBAR monitors the 5-minute bit rates of traffic types as well as the packet count and byte counts and from this information determines the types of traffic that need to be prioritized.

AutoQoS Implementation

AutoQoS in the Enterprise has some limitations, specifically on the types of interfaces that can be used with it. The following interfaces are the only ones that can be used with AutoQoS in the Enterprise:

Serial interfaces, PPP and HDLC encapsulation
Frame Relay interfaces, point-to-point only
ATM subinterfaces
Frame Relay to ATM internetworking links

There are also some restrictions on these types of interfaces. For PPP encapsulated serial interfaces, Multilink PPP (MLP) is automatically configured and the IP address on the interface is moved to the MLP bundle virtual interface. As well, Frame Relay DLCIs and ATM PVCs have some restrictions:

Frame Relay DLCI Restrictions
A Frame Relay map class cannot be associated with a DLCI.
A Frame Relay low speed link (<768 Kbps) cannot have a virtual template associated with a DLCI.
A Frame Relay to ATM internetworking link automatically runs MLP over Frame Relay and an IP address must be associated with the subinterface.

ATM PVC Restrictions
A virtual template cannot be associated with a low speed ATM PVC.
A low speed ATM PVC automatically runs MLP over ATM and an IP address must be associated with the subinterface.
When MLP over ATM is configured, the IP address is removed from the interface and put on the MLP bundle.

AutoQoS also has the ability to turn on different Cisco features which it believes will help in maintaining the priorities created with the policies. These features include Low-Latency Queuing (LLQ) and Weighted Round Robin (WRR) which are used for congestion management. Class-Based Traffic Shaping (CBTS) and Frame Relay Traffic Shaping (FRTS) controls the flow of traffic and thus raises the efficiency of the bandwidth available. Weighted Random Early Detection (WRED) is used for congestion avoidance and Link Fragmentation and Interleaving (LFI) and Compressed Real Time Protocol (cRTP) and MLP are used to increase link efficacy.

The following table shows the different QoS variables which are set by AutoQoS, AF = Assured Forwarding, CS = Class Selector, EF = Expedited Forwarding:

Layer 2 Class of Service	Layer 3 IP Precedence	DSCP	AutoQoS Class Name
CoS 0	Routine (IP precedence 0)	0-7	Best Effort
CoS 1	Priority (IP precedence 1)	8-15	Bulk Data (Web Traffic..)(AF11 – DSCP 10) Scavenger (CS1 – DSCP 8)
CoS 2	Immediate (IP precedence 2)	16-23	Transactional Databases (AF21 – DSCP 18) Network Management (CS2 – DSCP 16)
CoS 3	Flash (IP precedence 3)	24-31	Telephony Signaling
CoS 4	Flash-override (IP precedence 4)	32-39	Interactive Video (AF41 – DSCP 34) Streaming Video (CS4 –DSCP 32)
CoS 5	Critical (IP precedence 5)	40-47	Interactive Voice (EF – DSCP 46)
CoS 6	Internet (IP precedence 6)	48-55	IP Routing (CS6 – DSCP 48)
CoS 7	Network (IP precedence 7)	56-63	

Table 6 - AutoQoS Traffic Classes

AutoQoS (VoIP) and AutoQoS (Enterprise) Differences

There are two different types of AutoQoS which exist on the Cisco devices: AutoQoS for VoIP and AutoQoS for the Enterprise. AutoQoS for VoIP came out first and is used only to identify and provide QoS for VoIP traffic. AutoQoS for the Enterprise came out as a second version of AutoQoS and provides for the QoS of several different types of traffic (which are only limited by NBAR traffic identification). AutoQoS in the Enterprise adds an additional step which did not exist in the VoIP implementation. This is an autodiscovery phase which is used to monitor traffic over an amount of time to get a real picture of the traffic QoS needs (typically three days). Once AutoQoS gets a good picture of the QoS, needs it shapes a policy which will deal with the needs based on this discovery.

Configuration

policy-map

The **policy-map** *policy-map-name* command is used to create a policy map which will be used in conjunction with the **class-map** and **service-policy** commands. The *policy-map-name* parameter is used to assign a name to the policy. This name can be up to 40 alphanumeric characters.

Syntax:

```
router(config)#policy-map policy-map-name
```

class-map

The **class-map** *class-map-name* command is used to create a class map which is used in conjunction with the **policy-map** and **service-policy** commands. The **match** commands are used to specify which types of traffic are classified into a specific class.

Syntax:

```
router(config)#class-map class-map-name
```

class

The **class** [*class-name* | **class-default**] [**insert-before** *class-name*] command is used to create or change a class-map for a policy. The *class-name* parameter is used to specify the name of the class; the **class-default** parameter specifies the default class. The **insert-before** parameter is used to create a class-map between two existing class-maps. The *class-name* parameter is used to specify the class-map before which to insert the new class-map.

Syntax:

```
router(router-pmap)#class [class-name | class-default] [insert-before class-name]
```

service-policy input

The **service-policy input** *policy-map-name* command is used to attach a policy-map to an input interface or VC and is used in conjunction with the **policy-map** and **class-map** commands. The *policy-map-name* parameter is used to assign a name to the policy; this name can be up to 40 alphanumeric characters.

Syntax:

```
router(config-if)#service-policy input policy-map-name
```

service-policy output

The **service-policy output** *policy-map-name* command is used to attach a policy-map to an output interface or VC. It is used in conjunction with the **policy-map** and **class-map** commands. The *policy-map-name* parameter is used to assign a name to the policy; this name can be up to 40 alphanumeric characters.

Syntax:

```
router(config-if)#service-policy output policy-map-name
```

match protocol

The **match protocol** *protocol-name* command is used inside a class-map to match specific protocols. The *protocol-name* is the name of the protocol that is going to be matched. It can be a variety of protocols when using NBAR, including **citrix**, **dhcp**, **dns**, **eigrp**, **fastrack**, **gnutella**, **h323**, **http**, and **irc**, among many others.

Syntax:

```
router(config-cmap)#match protocol protocol-name
```

match fr-dlci

The **match fr-dlci** *dlci-number* command is used to match a specific frame relay DLCI number. The *dlci-number* parameter is the DLCI number which is to be matched.

Syntax:

```
router(config-cmap)#match fr-dlci dlci-number
```

match access group

The **match access-group** {*access-group* | **name** *access-group-name*} command is used to match a specific access list. The *access-group* and *access-group-name* parameters are used to reference a specific access-list.

Syntax:

```
router(config-cmap)#match access-group {access-group | name access-group-name}
```

match cos

The **match cos** *cos-value* [*cos-value* [*cos-value* [*cos-value*]]] command is used to match specific Layer 2 Class of Service (CoS) bits. The *cos-value* parameter is used to specify a Layer 2 CoS value. There can be several *cos-value* parameters defined to match multiple Layer 2 CoS values.

Syntax:

```
router(config-cmap)#match cos cos-value [cos-value [cos-value [cos-value]]]
```

match precedence

The **match [ip] precedence** *precedence-criteria1 precedence-criteria2 precedence-criteria3 precedence-criteria4* command is used to match specific Layer 3 precedence bits. The *precedence-criteria* is used to specify a Layer 3 precedence value. There can be several *precedence-criteria* parameters defined to match multiple Layer 3 precedence values.

Syntax:

```
router(config-cmap)#match [ip] precedence precedence-criteria1 precedence-criteria2  
precedence-criteria3 precedence-criteria4
```

match dscp

The **match [ip] dscp** *dscp-value* [*dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value*] commands are used to match the DSCP of the packet. The **ip** parameter is used to specify an IPv4 match. If this parameter is not used, a match with IPv4 or IPv6 is completed. The *dscp-value* parameter is used to specify the DSCP value to be matched. There can be several *dscp-value* parameters defined to match multiple DSCP values.

Syntax:

```
router(config-cmap)#match [ip] dscp dscp-value [dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value]
```

match input-interface

The **match input-interface** *interface-name* command is used to match the input interface of the packet. The *interface-name* parameter is the interface name of the intended match.

Syntax:

```
router(config-cmap)#match input-interface interface-name
```

match ip rtp

The **match ip rtp** *starting-port-number port-range* command is used to match a range of UDP even port numbers used by RTP. The *starting-port-number* parameter specifies the start of the port range to be matched. The *port-range* parameter specifies the end of the port range to be matched. This is done through adding the *starting-port-number* and the *port-range* number together to get the end of the range port number.

Syntax:

```
router(config-cmap)#match ip rtp starting-port-number port-range
```

match mpls experimental

The **match mpls experimental** *number* command is used to match a MPLS EXP value. The *number* parameter is used to specify the MPLS EXP value that is to be matched. There can be several *number* parameters defined to match multiple MPLS EXP values.

Syntax:

```
router(config-cmap)#match mpls experimental number
```

match mpls experimental topmost

The **match mpls experimental topmost** *number* command is used to match the topmost MPLS EXP value. The *number* parameter is used to specify the topmost MPLS EXP value that is to be matched.

Syntax:

```
router(config-cmap)#match mpls experimental topmost number
```

match packet length

The **match packet length** [**min** *minimum-length-value*] [**max** *maximum-length-value*] command is used to match the length field in the IP packet. The **min** parameter is used to specify the minimum length of the packet. The **max** parameter is used to specify the maximum length of the packet. If only the **min** parameter is specified, then all values above the **min** parameter are matched. If only the **max** parameter is specified, all values above the **max** parameter are matched.

Syntax:

```
router(config-cmap)#match packet length [min minimum-length-value] [max maximum-length-value]
```

set atm-clp

The **set atm-clp** command is used within a policy-map to set the ATM CLP bit.

Syntax:

```
router(config-pmap-c)#set atm-clp
```

set cos

The **set cos** *cos-value* command is used within a policy-map to set the Layer 2 CoS value of the outgoing packet. The *cos-value* parameter specifies the CoS value to be set.

Syntax:

```
router(config-pmap-c)#set cos cos-value
```

set precedence

The **set precedence** *precedence-value* command is used within a policy-map to set the IP precedence value. The *precedence-value* parameter specifies the IP precedence value to be set.

Syntax:

```
router(config-pmap-c)#set precedence precedence-value
```

set dscp

The **set [ip] dscp** *dscp-value* command is used within a policy-map to set the DSCP value. The **ip** parameter is used to specify an IPv4 match. If this parameter is not used, a match with IPv4 or IPv6 is completed. The *dscp-value* parameter specifies the DSCP value to be set.

Syntax:

```
router(config-pmap-c)#set [ip] dscp dscp-value
```


set fr-de

The **set fr-de** command is used within a policy-map to set the Frame-Relay DE bit.

Syntax:

```
router(config-pmap-c)#set fr-de
```

set ip tos

The **set ip tos** [*tos-bit-value* | **max-reliability** | **max-throughput** | **min-delay** | **min-monetary-cost** | **normal**] command is used within a policy-map to set the ToS bits. The *tos-bit-value* parameter specifies the ToS value to be set from 0 to 15. The **max-reliability** parameter sets the ToS value to 2. The **max-throughput** parameter sets the ToS value to 4. The **min-delay** parameter sets the ToS value to 8. The **min-monetary-cost** parameter sets the ToS value to 1. The **normal** parameter sets the ToS to 0.

Syntax:

```
router(config-pmap-c)#set ip tos [tos-bit-value | max-reliability | max-throughput | min-delay | min-monetary-cost | normal]
```

set mpls experimental imposition

The **set mpls experimental imposition** *mpls-exp-value* command is used within a policy-map to set the MPLS EXP value. The *mpls-exp-value* parameter sets the MPLS EXP value. This value is from 0 to 7.

Syntax:

```
router(config-pmap-c)#set mpls experimental imposition mpls-exp-value
```

set mpls experimental topmost

The **set mpls experimental topmost** *mpls-exp-value* command is used within a policy-map to set the topmost MPLS EXP value. The *mpls-exp-value* parameter sets the topmost MPLS EXP value. This value is from 0 to 7.

Syntax:

```
router(config-pmap-c)#set mpls experimental topmost mpls-exp-value
```

bandwidth (CBWFQ)

The **bandwidth** {*bandwidth-kbps* | **remaining percent** *percentage* | **percent** *percentage*} command is used to specify the amount of bandwidth specified to a class inside a policy-map for CBWFQ. The *bandwidth-kbps* parameter specifies the amount of bandwidth to be assigned to the class. The **remaining percent** *percentage* parameter is used to specify a percentage of bandwidth based on the relative amount of remaining bandwidth. The **percent** *percentage* parameter is used to specify a percentage of bandwidth based on the absolute percentage.

Syntax:

```
router(config-pmap-c)#bandwidth {bandwidth-kbps | remaining percent percentage | percent percentage}
```

random-detect

The **random-detect** [**dscp-based** | **prec-based**] command is used to enable WRED on an interface or class of a policy-map. The **dscp-based** parameter is used to specify the use of DSCP values. The **prec-based** parameter is used to specify the use of IP precedence values.

Syntax:

```
router(config-if)#random-detect [dscp-based | prec-based]
```

or

```
router(config-pmap-c)#random-detect [dscp-based | prec-based]
```

random-detect dscp

The **random-detect dscp** *dscp-value min-threshold max-threshold [max-probability-denominator]* command is used to set the minimum and maximum thresholds for DSCP values. The *dscp-value* parameter specifies the DSCP value to be configured. This can be a number from 0 to 63 or the keywords **af11**, **af12**, **af13**, **af21**, **af22**, **af23**, **af31**, **af32**, **af33**, **af41**, **af42**, **af43**, **cs1**, **cs2**, **cs3**, **cs4**, **cs5**, **cs7**, **ef**, or **rsvp**. The *min-threshold* parameter specifies the minimum threshold for average queue length to start to be dropped in number of packets. This can be from 1 to 4096. The *max-threshold* specifies the maximum threshold for average queue length to be completely dropped in number of packets. This can be from 1 to 4096. The *max-probability-denominator* parameter is used to specify the probability denominator. This is used to show the amount of packets which are dropped when at maximum threshold (but not over). The denominator is used as a fraction of packets. The default is 10 on most Cisco equipment, which means one packet in ten will be dropped when at the maximum threshold.

Syntax:

```
router(config-if)#random-detect dscp dscp-value min-threshold max-threshold [max-probability-denominator]
```

or

```
router(config-pmap-c)#random-detect dscp dscp-value min-threshold max-threshold [max-probability-denominator]
```

random-detect precedence

The **random-detect precedence** *{precedence | rsvp} min-threshold max-threshold max-probability-denominator* command is used to set the minimum and maximum thresholds for IP precedence values. The *precedence* parameter specifies the IP precedence value to be configured; this can be a number from 0 to 7. The **rsvp** parameter is used to specify RSVP traffic. The *min-threshold* parameter specifies the minimum threshold for average queue length to start to be dropped in number of packets. This can be from 1 to 4096. The *max-threshold* specifies the maximum threshold for average queue length to be completely dropped in number of packets; this can be from 1 to 4096. The *max-probability-denominator* parameter is used to specify the probability denominator. This is used to show the amount of packets which are dropped when at maximum threshold (but not over). The denominator is used as a fraction of packets. The default is 10 on most Cisco equipment, which means one packet in ten will be dropped when at the maximum threshold.

Syntax:

```
router(config-if)#random-detect precedence {precedence | rsvp} min-threshold max-threshold max-probability-denominator
```

or

```
router(config-pmap-c)#random-detect precedence {precedence | rsvp} min-threshold max-threshold max-probability-denominator
```

random-detect exponential-weighting-constant

The **random-detect exponential-weighting-constant** *exponent* command is used to specify the exponential weight factor used in average queue calculations. The *exponent* parameter is used to specify the exponent to be used. By default this value is 9, but can be from 1 to 16.

Syntax:

```
router(config-if)#random-detect exponential-weighting-constant exponent
```

or

```
router(config-pmap-c)#random-detect exponential-weighting-constant exponent
```

compression header ip

The **compression header ip** *[rtp | tcp]* command is used to enable IP header compression. The **rtp** parameter is used to enable RTP header compression. The **tcp** parameter is used to enable TCP header compression.

Syntax:

```
router(config-pmap-c)#compression header ip [rtp | tcp]
```

auto discovery qos

The **auto discovery qos [trust]** command is used to enable AutoQoS for the Enterprise autodiscovery. This must be done before enabling AutoQoS on the interface and will not be recalculated once complete unless invoked again. The **trust** parameter tells the discovery process to trust QoS values which are already inserted into the traffic. This is typically used if other devices in the network are marking traffic and these devices are trusted.

Syntax:

```
router(config-if)#auto discovery qos [trust]
```

auto qos

The **auto qos** command is used to install AutoQoS in the Enterprise class-maps and policy-maps to an interface.

Syntax:

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

auto qos voip

The **auto qos voip [trust]** command is used to install AutoQoS (VoIP) class-maps and policy-maps to an interface. The **trust** parameter tells the AutoQoS to trust QoS values which are already inserted into the traffic. This is typically used if other devices in the network are marking traffic and these devices are trusted.

Syntax:

```
router(config-if)#auto qos voip [trust]
```

auto qos voip cisco-phone

The **auto qos voip cisco-phone** command is used to enable AutoQoS (VoIP) on an interface which is connected to a Cisco IP phone. This feature is available on Cisco switches and configures the QoS parameters based on the values marked by the phone. It requires CDP version 2, which is used by the switch to verify a Cisco phone is attached.

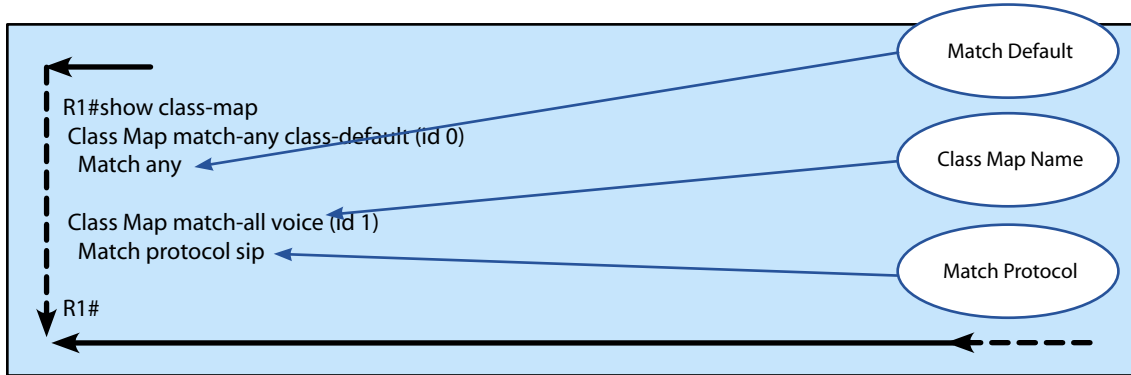
Syntax:

```
router(config-if)#auto qos voip cisco-phone
```

Troubleshooting

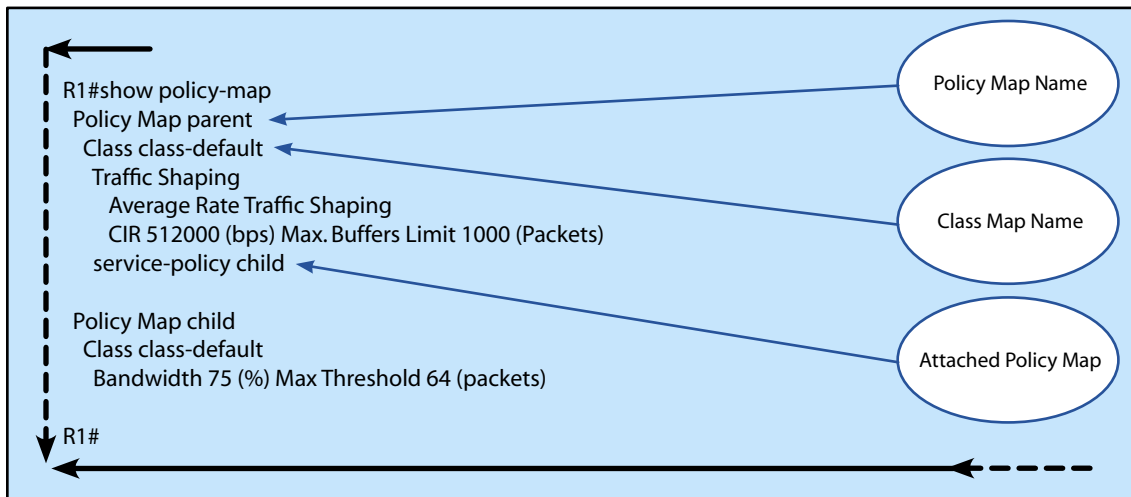
show class-map

This command is used to display information about configured class-maps. The following highlights the most important parts.



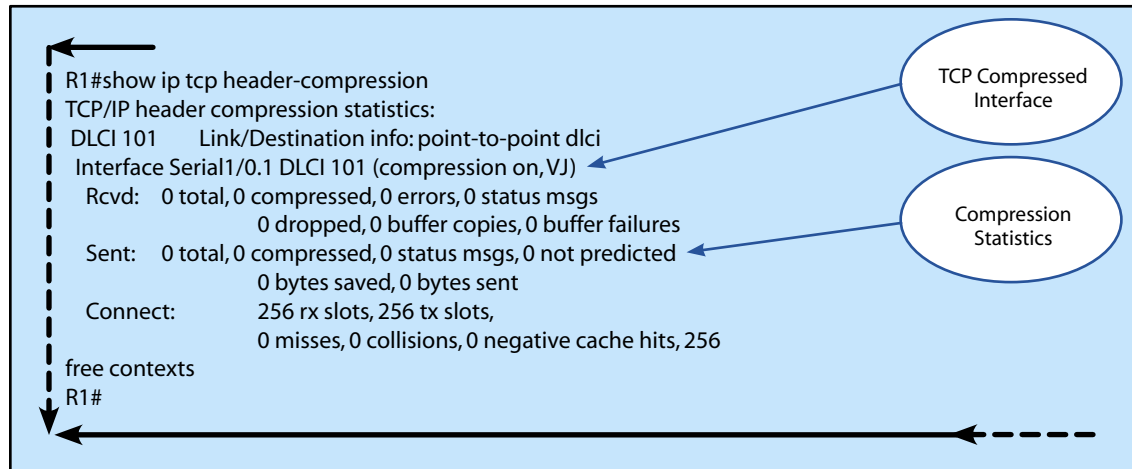
show policy-map

This command is used to display information about configured policy-maps. The following highlights the most important parts.



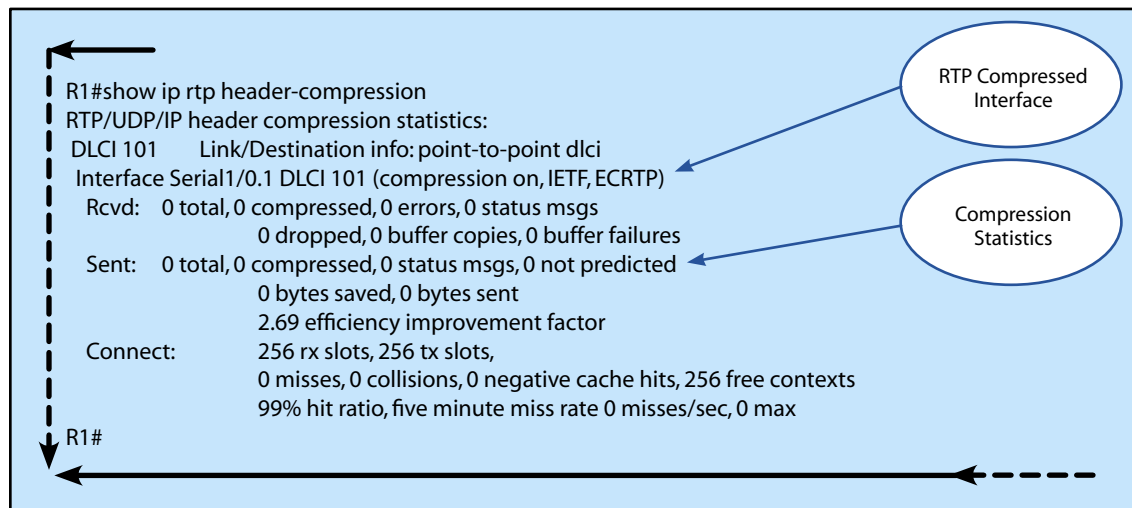
show ip tcp header-compression

This command is used to display information about IP TCP header compression statistics. The following highlights the most important parts.



show ip rtp header-compression

This command is used to display information about IP RTP header compression statistics. The following highlights the most important parts.



show auto discovery qos

This command is used to display the traffic classes that have been found by AutoQoS autodiscovery. The following highlights the most important parts.

```

R1#show auto discovery qos
FastEthernet0/0
AutoQoS Discovery enabled for applications
Discovery up time: 42 minutes, 40 seconds
AutoQoS Class information:
Class Voice:
No data found.
Class Interactive Video:
No data found.
Class Signaling:
No data found.
Class Streaming Video:
No data found.
Class Transactional:
No data found.
Class Bulk:
No data found.
Class Scavenger:
No data found.
Class Management:
No data found.
Class Routing:
No data found.
Class Best Effort:
Current Bandwidth Estimation: 3 Kbps/<1% (AverageRate)
Detected applications and data:
Application/   AverageRate   PeakRate   Total
Protocol      (kbps/%)     (kbps/%)   (bytes)
-----
http          3/<1          94/<1      1031164
unknowns     0/0           1/<1       60300

Suggested AutoQoS Policy for the current uptime:
!
policy-map AutoQoS-Policy-Fa0/0
class class-default
fair-queue
R1#
  
```

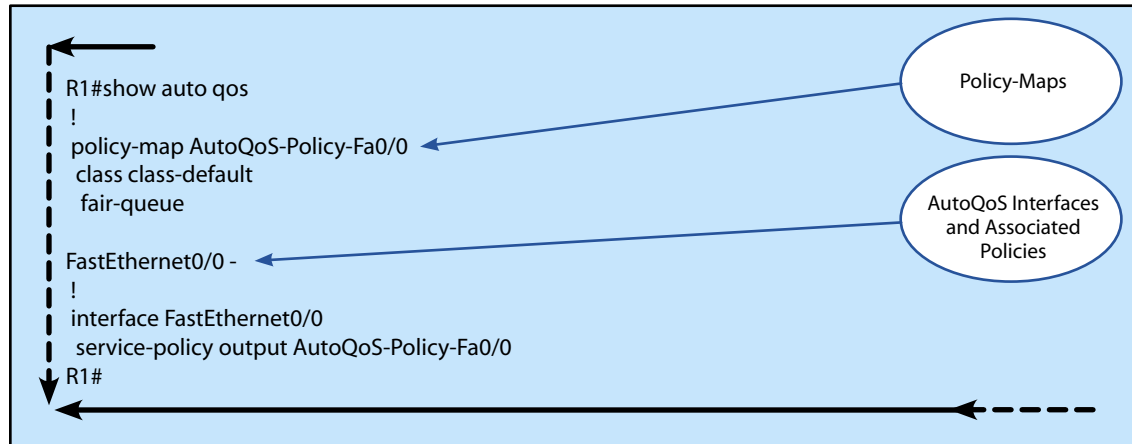
The diagram highlights the following parts of the output:

- Interface:** Points to the interface name `FastEthernet0/0`.
- AutoQoS Classes:** Points to the `AutoQoS Class information:` section.
- Best Effort Traffic Found:** Points to the table of detected applications and data.
- Policy Recommendations:** Points to the `policy-map AutoQoS-Policy-Fa0/0` section.

Application/ Protocol	AverageRate (kbps/%)	PeakRate (kbps/%)	Total (bytes)
http	3/<1	94/<1	1031164
unknowns	0/0	1/<1	60300

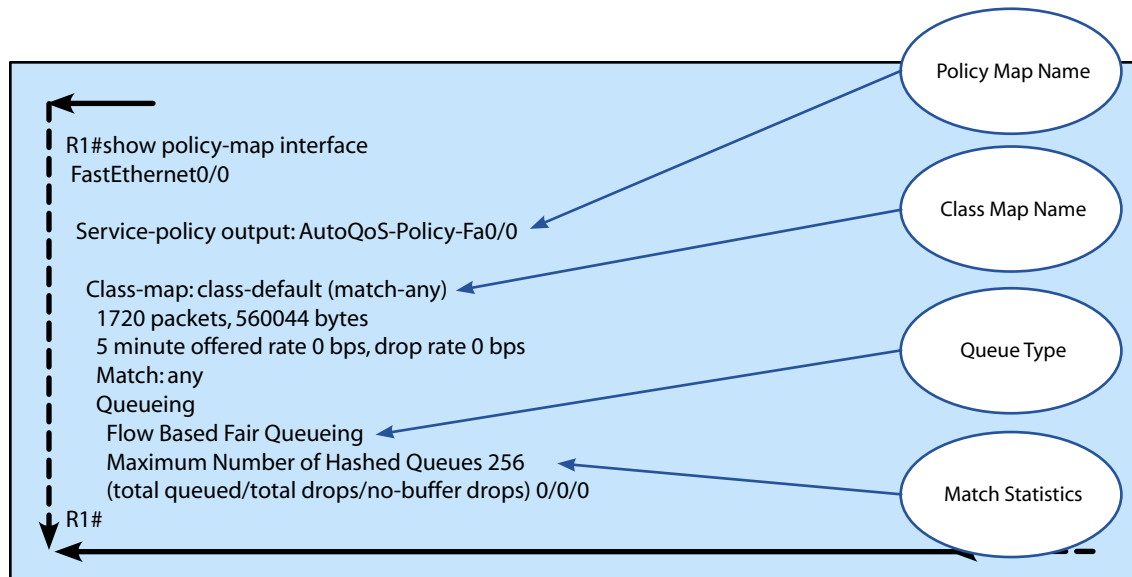
show auto qos

This command is used to display information about AutoQoS configured policies and interfaces. The following highlights the most important parts.



show policy-map interface

This command is used to display information about the policy-map configuration on an interface. The following highlights the most important parts.



Domain 4 - Classification and Marking

Classification and Marking Basics

Marking is the action of changing a part of the traffic to mark its classification so that networking equipment along the path can route based on this classification. This can be done in a number of ways because there are a number of different options for marking. Domain two detailed these markings, but these will be reviewed briefly. At layer 2, there are Class of Service (CoS) bits which enable marking with Ethernet (802.1Q/P). With frame relay, there is a Discard Eligible (DE) bit. With Asynchronous Transfer Mode (ATM) there is a Cell Loss Priority (CLP) bit, which enables the marking of priority and non-priority traffic. At Layer 2 ½ with Multiprotocol Label Switching (MPLS), there is an EXP field (3-bits) which is meant to work in conjunction with the layer 3 Type of Service (ToS) field (first three bits of ToS). At layer 3, there is the IP ToS (8-bit) field. This field can be a little confusing because it has multiple implementations. The original RFC specified the first 3 bits of the ToS field to be for IP precedence and the 4th, 5th and 6th bits to be used to request low delay, high throughput and high reliability respectively. The new RFC uses the ToS field in a different way. This provides the definition of the Differentiated Services (DS or DiffServ) within the DS field is the 6-bit Differentiated Service Codepoint (DSCP) and the 2-bit Explicit Congestion Notification (ECN).

On Cisco equipment the following are the recommended settings for CoS, IP Precedence or DSCP:

Traffic Type	CoS	IP Precedence	DSCP
Voice Payload	5	5	EF
Video Payload	4	4	AF41
Voice/Video Signaling	3	3	CS3
Mission Critical Data	3	3	AF31 AF32 AF33
Transactional Data	2	2	AF21 AF22 AF23
Bulk Traffic	1	1	AF11 AF12 AF13
Best Effort	0	0	BE
Scavenger	0	0	2 4 5

Table 7 - Cisco Recommended Settings

Classification and Marking Configuration

The classification and marking configuration on Cisco equipment is also done through the use of MQC. This means that if it is known how the **class-map**, **policy-map** and **service-map** commands work anything on the equipment can be classified and marked. This means that traffic can be classified based on a number of criteria as shown in domain three. If interested in classification examples based on DSCP, IP Precedence or CoS, please review the examples in domain three.

Network Based Application Recognition (NBAR)

The use of Network Based Application Recognition (NBAR) was not discussed in Domain 3. NBAR is a traffic classification engine which can be used for multiple tasks in a QoS configuration. NBAR is meant to be used as an alternative to manually using access lists for classification, which can be quite configuration intensive. NBAR provides the ability to classify traffic based on static TCP/UDP port numbers, Non-TCP/UDP traffic, and dynamically assigned TCP/UDP ports. It also provides the ability to perform deep packet inspection which provides sub-port classification. Sub-port classification allows the networking equipment to inspect the traffic from layers 4 through layer 7. This includes URL inspection (HTTP), Multipurpose Internet Mail Extension (MIME) inspection, Citrix traffic inspection, and RTP traffic inspection, as well as the ability to create custom NBAR application templates.

NBAR also has the ability to have additional traffic matching modules added which can be downloaded from Cisco. These modules are called Packet Description Language Modules (PDLM). NBAR has some limitations which need to be considered when using its capabilities.

NBAR Restrictions
Limited to a total of 24 concurrent URLs, hosts, or MIME type matches
Limited to first 400 bytes of the traffic prior to IOS 12.3(7)T. Post IOS 12.3(7)T has the ability to perform full packet inspection.
Custom templates are limited to the first 255 bytes of the payload.
Limited to only IP traffic, this includes not being able to inspect MPLS traffic
No multicast inspection
Limited to CEF switching mode
No fragmented packets
No SSL-HTTP packet inspection
Limited to traffic not originating in or destined for the networking equipment performing NBAR
Not supported on Fast Etherchannel interfaces or interfaces using tunneling or encryption

NBAR Configuration

NBAR can be used in a variety of different configuration options, from simple matching of protocols for simple classification to its use for classification of traffic for advanced queuing techniques. The examples below will show how NBAR can be used in these different configurations.

Basic NBAR Configuration

The following is a simple configuration which is used to match SIP and HTTP traffic and assign this traffic with DSCP. All other traffic that is not matched is assigned a default DSCP value.

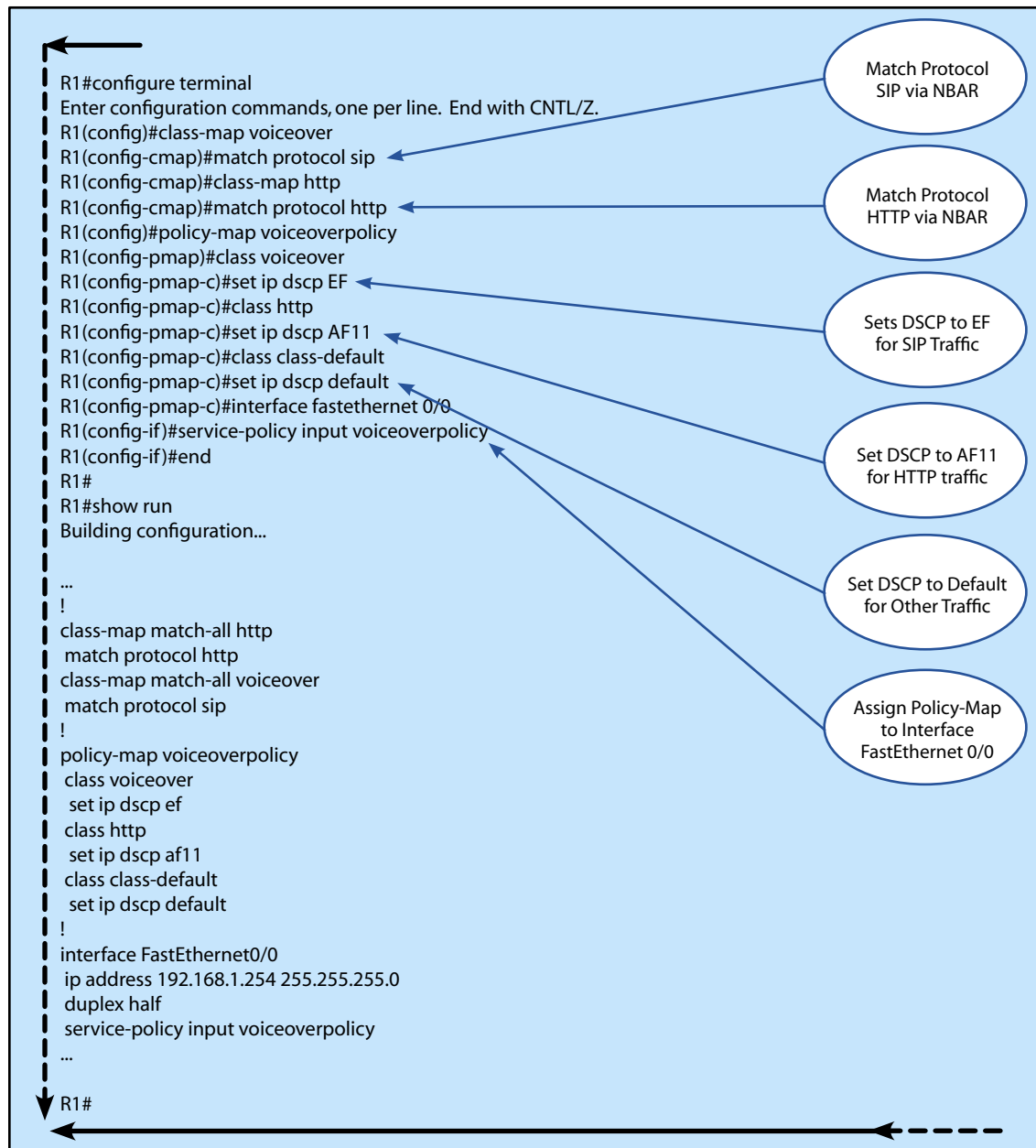


Figure 23 - MQC NBAR Example

Basic Classification and Marking Configuration

One of the fundamental purposes of classification and marking is to put each different type of traffic into its appropriate class. This enables it to be acted upon based on its priority against other traffic. Examples of this have been shown earlier in this manual. Another part of classification and marking that must happen is the translation of different types of marking. To review, the three main marking types are CoS, IP Precedence and DSCP. The following examples show how this can be done.

CoS to DSCP Configuration

The following example shows one of the ways for classification and marking to translate CoS markings into DSCP markings.

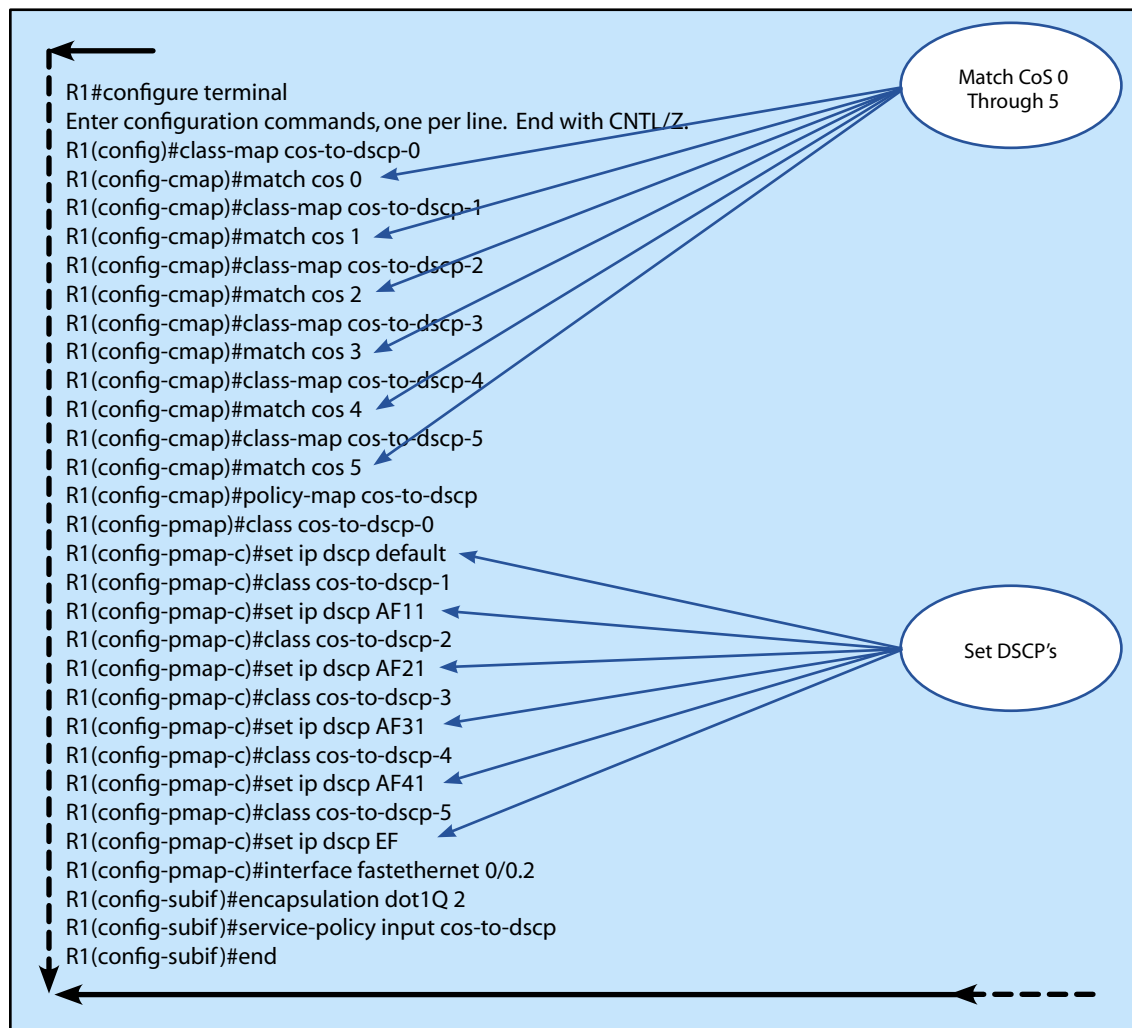


Figure 24 - CoS to DSCP Conversion Example

```
R1#show running-config
Building configuration...

Current configuration : 2341 bytes
...
!
class-map match-all cos-to-dscp-5
match cos 5
class-map match-all cos-to-dscp-4
match cos 4
class-map match-all cos-to-dscp-1
match cos 1
class-map match-all cos-to-dscp-0
match cos 0
class-map match-all cos-to-dscp-3
match cos 3
class-map match-all cos-to-dscp-2
match cos 2
!
!
policy-map cos-to-dscp
class cos-to-dscp-0
set ip dscp default
class cos-to-dscp-1
set ip dscp af11
class cos-to-dscp-2
set ip dscp af21
class cos-to-dscp-3
set ip dscp af31
class cos-to-dscp-4
set ip dscp af41
class cos-to-dscp-5
set ip dscp ef
!
!
interface FastEthernet0/0
ip address 192.168.1.254 255.255.255.0
duplex half
!
interface FastEthernet0/0.2
encapsulation dot1Q 2
service-policy input cos-to-dscp
!
...
R1#
```

Figure 25 - CoS to DSCP Conversion Running Configuration

IP Precedence to DSCP Configuration

The following example shows one of the ways for classification and marking to translate IP Precedence markings into DSCP markings.

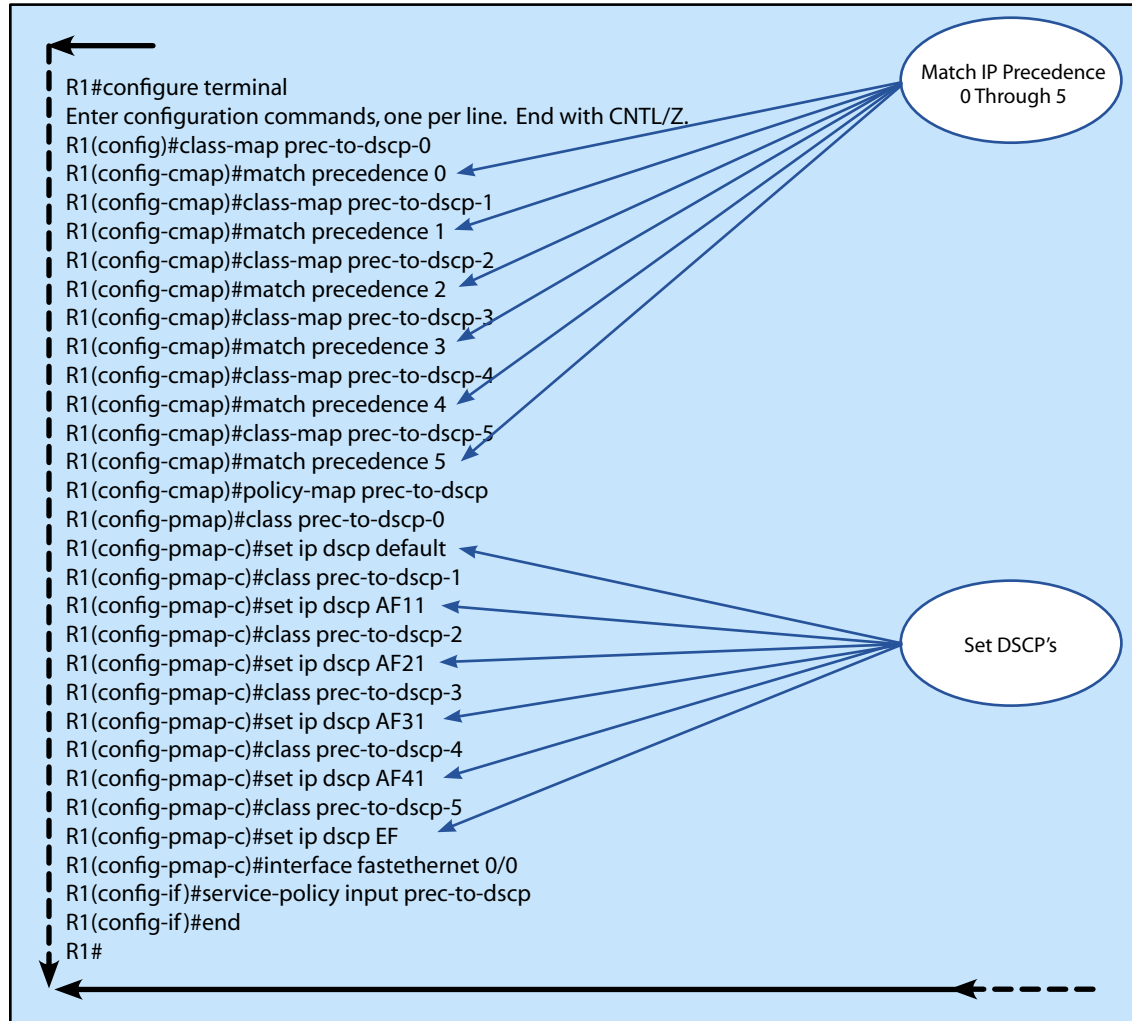


Figure 26 - IP Precedence to DSCP Conversion Example

```
R1#show running-config
Building configuration...
...
!
class-map match-all prec-to-dscp-4
 match precedence 4
class-map match-all prec-to-dscp-5
 match precedence 5
class-map match-all prec-to-dscp-2
 match precedence 2
class-map match-all prec-to-dscp-3
 match precedence 3
class-map match-all prec-to-dscp-0
 match precedence 0
class-map match-all prec-to-dscp-1
 match precedence 1
!
policy-map prec-to-dscp
 class prec-to-dscp-0
  set ip dscp default
 class prec-to-dscp-1
  set ip dscp af11
 class prec-to-dscp-2
  set ip dscp af21
 class prec-to-dscp-3
  set ip dscp af31
 class prec-to-dscp-4
  set ip dscp af41
 class prec-to-dscp-5
  set ip dscp ef
!
interface FastEthernet0/0
 ip address 192.168.1.254 255.255.255.0
 duplex half
 service-policy input prec-to-dscp
...
R1#
```

Figure 27 - IP Precedence to DSCP Conversion Running Configuration

Pre-Classify

QoS Pre-Classify is used only on tunnel interfaces, virtual templates and crypto maps because it is not needed in other circumstances. What Pre-Classify does is provide the ability to correctly classify traffic that has been tunneled and/or encrypted for transmission. When this happens and a service-map exists on the interface, the tunneled or encrypted packets are classified based on the post-tunneled or post-encrypted IP packets. Now, if classification is based solely on the ToS byte, then there will be no problem as the ToS byte of the original packet is *automatically* put onto the tunneled or encrypted one. However, if classification is based on any other parameter like IP address, TCP/UDP port or the protocol number, then there will be a problem because these are all replaced with the information for the tunnel or encrypted destination. Pre-Classify allows the service map to classify based on the *original* IP packet headers and not the new or *tunneled* ones. In order to do this, the **qos pre-classify** command must be used on the interface.

One important thing to point out is where to use the **qos pre-classify** command. For GRE and IPsec VPNs, put the **qos pre-classify** command on the tunnel interface. For L2F and L2TP VPNs, put the **qos pre-classify** command on the virtual-template interface. For IPsec VPNs, put the **qos pre-classify** command on the crypto map.

Trust Boundaries

Trust boundaries are setup within a network that uses QoS. Specifically, a trust boundary is used to set the point in the network where the markings of incoming traffic are trusted and not re-classified. This can be setup at any point in the network, but it is recommended that the trust boundary point be as close to the end system as possible. Normal end systems are typically not trusted with the exception of IP phones, but the access switch or access router is a good place to start should the equipment have the ability and processor power to classify and mark the traffic in accordance with the network policies. This trust boundary is configured simply according to where the classification and marking on the network is done. At that point, it can be delineated as to how the different markings are treated and thus trust is obtained or refused.

Configuration

ip nbar protocol-discovery

The **ip nbar protocol-discovery** command is used to enable NBAR discovery on an interface.

Syntax:

```
router(config-if)#ip nbar protocol-discovery
```

ip nbar pdlm

The **ip nbar pdlm** *pdlm-name* command is used to extend the number of protocols discovered by NBAR. The *pdlm-name* parameter specifies the url of the Packet Description Language Module (PLDM) on the flash card.

Syntax:

```
router(config)#ip nbar pdlm pdlm-name
```


ip nbar custom

The **ip nbar custom** *name* [*offset* [*format value*]] [*source* | *destination*] [**tcp** | **udp**] [**range** *start end* | *port-number*] command is used to extend the capabilities of NBAR. The *name* parameter is used to specify the name of the custom protocol that is being matched. The *offset* parameter is used to specify the position offset relative to the end of the IP header to begin inspecting. The *format* parameter is used to specify the format of the *value* parameter; this can be set to **ascii**, **hex**, or **decimal**. The *value* parameter specifies the value to be searching for in the packet, based on the *offset* and *format* parameters. The **tcp** and **udp** parameters specify optionally whether to look at only TCP or UDP traffic. The **range** parameter is used to specify a range of ports to be monitored. The range is assigned with the *start* and *end* parameters or the *port-number* parameter.

Syntax:

```
router(config)#ip nbar custom name [offset [format value]] [source | destination] [tcp |  
udp] [range start end | port-number]
```

qos pre-classify

The **qos pre-classify** command is used to enable QoS pre-qualify.

Syntax:

```
router(config-if)#qos pre-classify
```

or

```
router(config-crypto-map)#qos pre-classify
```

Domain 5 - Congestion Management Methods

Congestion Management Mechanisms

Congestion on a network is a simple concept. If the network has more traffic trying to be sent than capacity, congestion occurs. In order to manage congestion, a number of queuing mechanisms exist to try to deal with different types of congestion and different requirements for the traffic being sent during congested times. Another way to deal with congestion is to implement congestion avoidance, which works by selectively dropping traffic to try to avoid congestion. In the following sections, these different types of congestion management and congestion avoidance mechanisms will be discussed.

First In, First Out (FIFO) Queuing

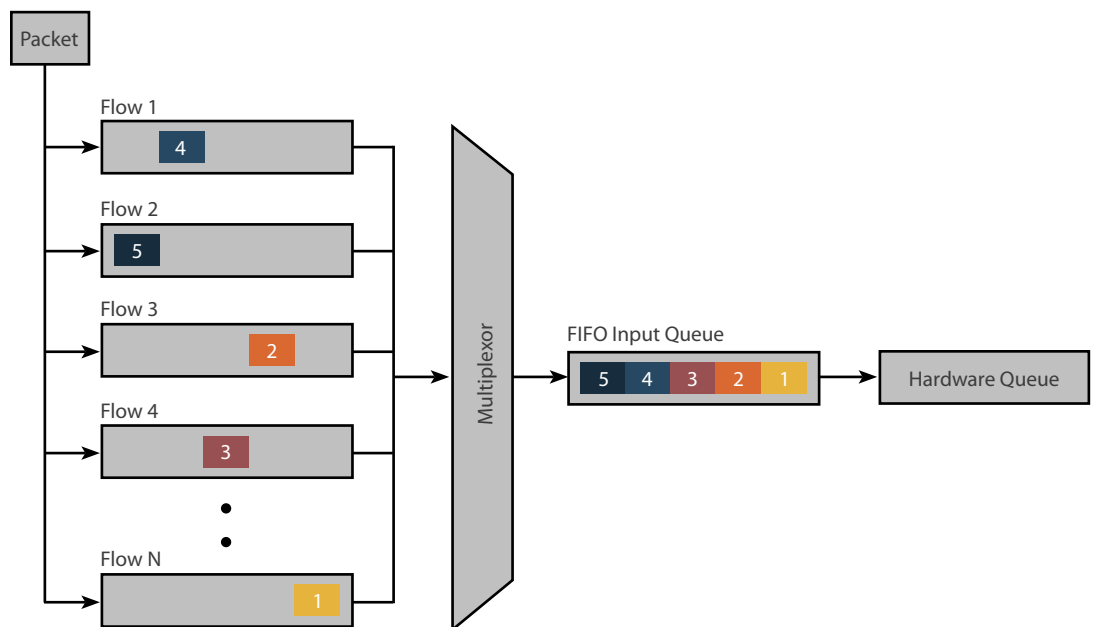


Figure 28 - FIFO Queuing Example

FIFO is the simplest of the queuing mechanisms. It requires no configuration and is very simple to understand. FIFO has a queue which is setup with an entrance and an exit. All traffic, without regard for priority, enters the queue and works its way through the queue. The traffic that first entered the queue is the first to get out of the queue. Since FIFO gives preference to no traffic, traffic flows that require large amounts of traffic monopolize the bandwidth. FIFO queuing is typically used on large bandwidth, low delay links and is considered the fastest queuing mechanism.

Priority Queuing (PQ)

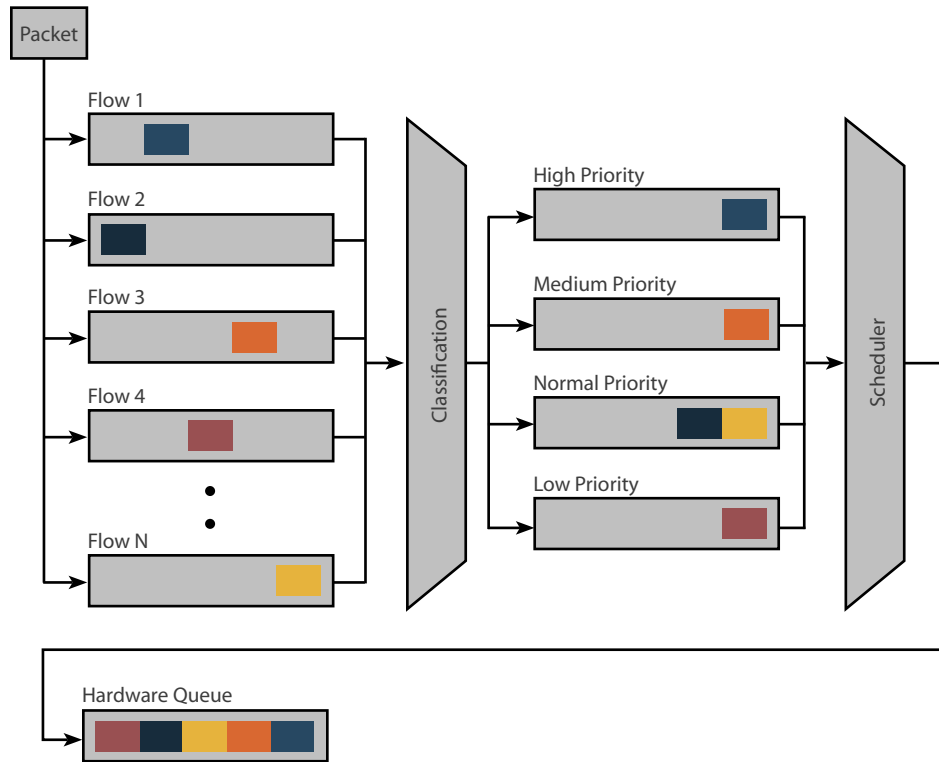


Figure 29 - Priority Queuing Example

Priority queuing works by prioritizing traffic into four different queues. This is typically done through the use of access lists and the **priority-list** command. The four queues which are used are high, medium, normal and low priority in order. The problem that exists with priority queues is that each of the higher queues is always serviced completely before the lower priority queues. This means that if there is a steady flow of higher priority traffic, the lower queues will fill and then start to tail drop packets.

Weighted Round Robin (WRR) and Custom Queuing (CQ)

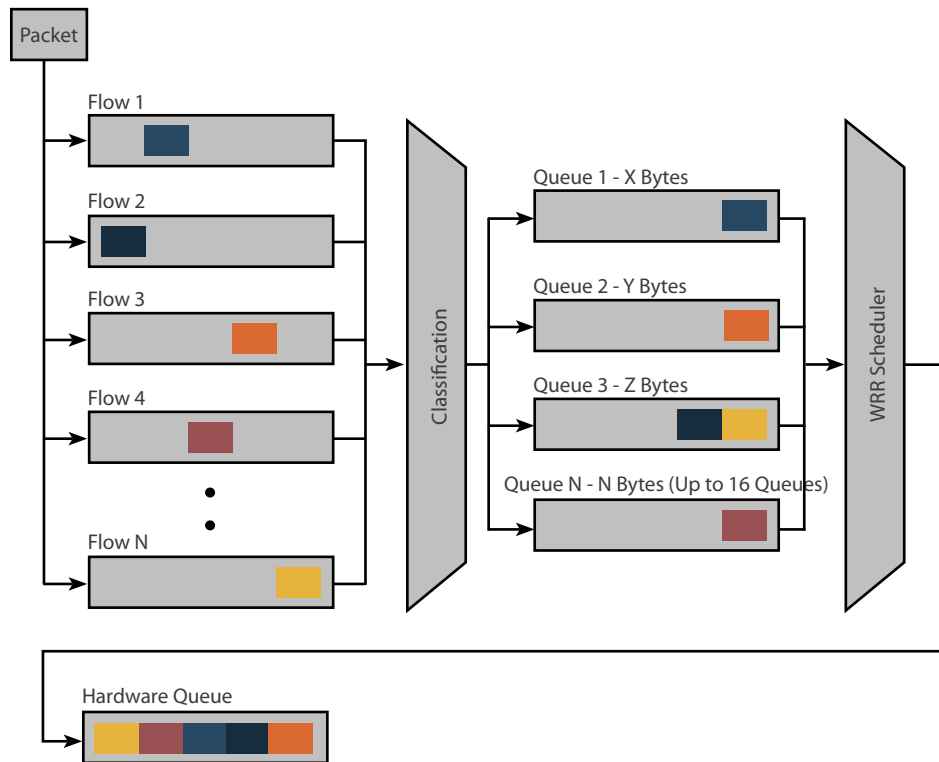


Figure 30 - Custom Queuing (WRR) Example

Generic round robin is a simple concept. It goes from queue to queue and each queue is allowed to send one packet before moving on to the next queue in line. With weighted round robin, the basic concept is the same, but each queue is allotted a weight which allows a queue to be serviced longer than the others. Within the Cisco world, custom queuing uses this model. It does this by allowing a specified amount of traffic to be configured which is serviced by each queue before moving on to the next queue. This traffic is measured in bytes. Cisco allows for 16 total custom queues (Queue 1 through Queue 16) and a system queue (Queue 0) which is used for keepalives and signaling traffic. Each queue is serviced until meeting or exceeding the byte count, then moves on to the next queue.

Weighted Fair Queuing (WFQ)

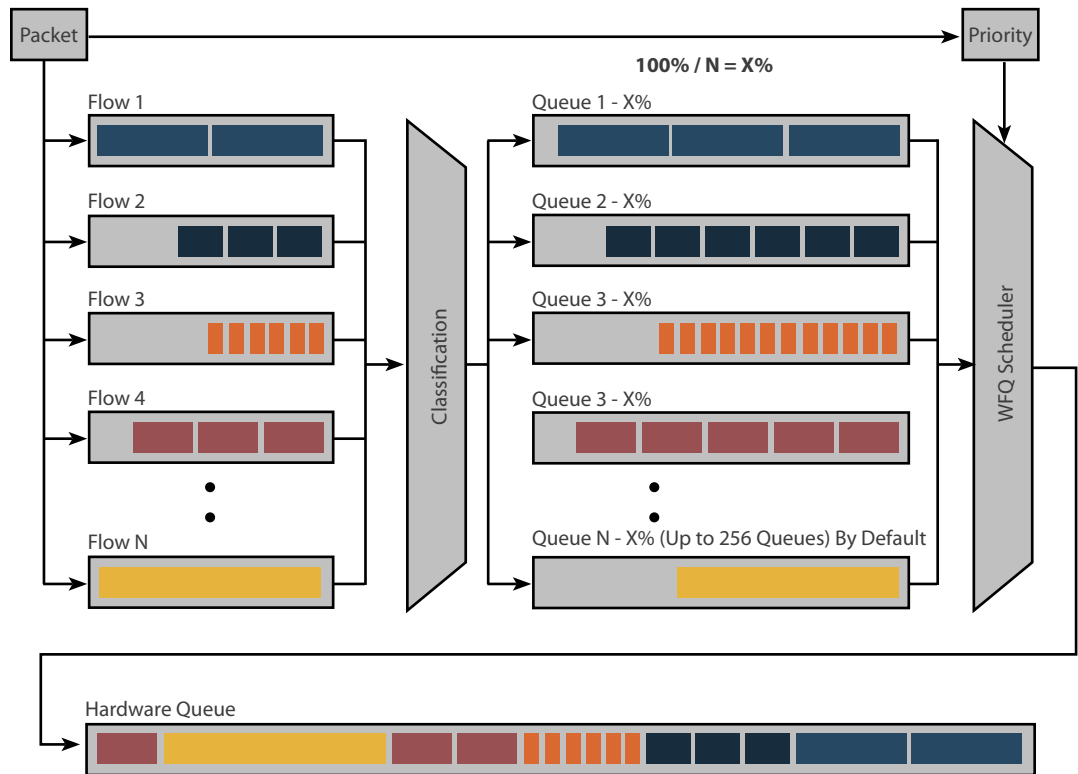


Figure 31 - Weighted Fair Queuing Example

WFQ is also called flow-based WFQ. It is called this because of what it does first, which is organize the traffic coming into it into flows. The ability to delineate one flow from another is done through the use of a hashing mechanism. This hash is created for all traffic coming into the WFQ interface and is created from the following:

Source IP Address
Destination IP Address
Protocol Number
Type of Service
Source TCP/UDP port number
Destination TCP/UDP port number

Once this hash has been created for the traffic, it is compared to the current queues that exist to determine if there was earlier traffic with the same hash. If so, the traffic is placed in the same queue and if not a new queue is created for each different hashed flow. By default, there are 256 different queues (also called *dynamic queues*) for the traffic to be placed in, including up to eight queues for system traffic and up to 1000 queues for RSVP traffic. There is a maximum configurable queue amount of 4096. However, the maximum queue amount must be a power of 2 (i.e. 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096).

WFQ has the advantage of simplicity as there is no required configuration. WFQ is the default on all serial interfaces with 2.048 Mbps (E1) bandwidths and below. It also guarantees that none of the flows will get starved for bandwidth as in some of the other queuing mechanisms. However, one of the main limitations to WFQ is that the classification and scheduling are not configurable. This is addressed in the next section.

WFQ is also IP precedence aware, meaning that it has the ability to give each flow an amount of preference should the IP precedence bits be set. This is done through a weighting mechanism. This mechanism uses the following formula with the lower numbers getting preference:

$$\frac{32384}{\text{IP Precedence} + 1}$$

Figure 32 - WFQ Weight Formula

The effects of IP precedence are shown in the following example:

If there are three flows, one with an IP precedence of 1, one with an IP precedence of 3, and one with an IP precedence of 5, WFQ would effectively do the following with the scheduled packets:

$$1 + 3 + 5 = 8$$

So the packets with an IP precedence of 1 would get $\frac{1}{8}$ the schedule, the packets with an IP precedence of 3 would get $\frac{3}{8}$ of the schedule, and the packets with an IP precedence of 5 would get $\frac{5}{8}$ of the schedule.

WFQ is also RSVP aware and will allocate reserved queue space for RSVP flows.

Class Based – Weighted Fair Queuing (CBWFQ)

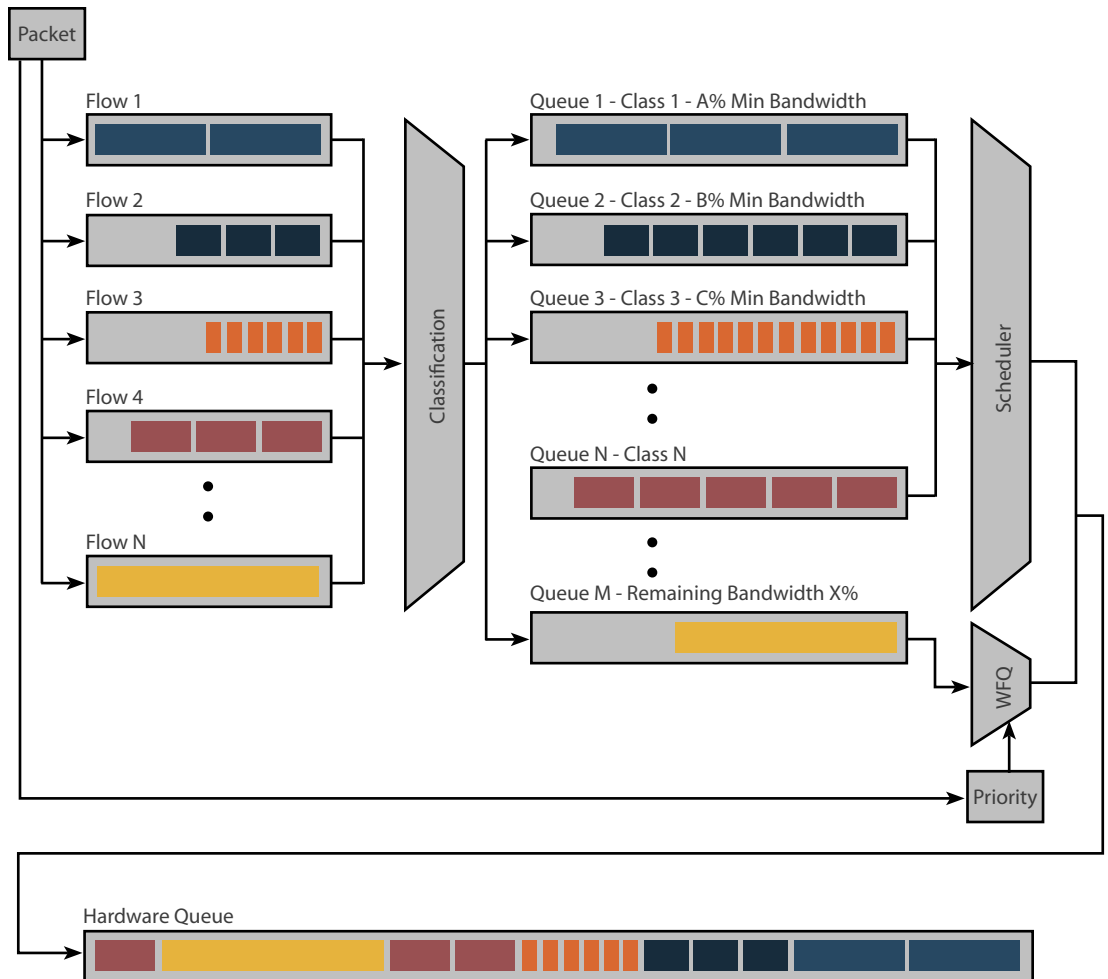


Figure 33 - Class Based - WFQ Queueing Example

CBWFQ is a new queuing mechanism which includes a couple of advantages from other queuing mechanisms. It allows the creation of classes which can be used to set either a minimum bandwidth allowed or a minimum percentage of the interface bandwidth. This guarantee is unique to CBWFQ. CBWFQ also provides this ability without the chance of starving any individual queue. CBWFQ also gives the ability to classify traffic with the MQC class-maps, which is considerably easier than the advanced access lists required with priority queuing. These queues, which are created from the class-maps, are serviced based on the minimum bandwidth requirements in a fair-queue way, meaning that each queue is serviced enough to meet the bandwidth requirements when it moves to the next queue and so on.

CBWFQ allows for up to 64 queues to be created for user classes. Each of these queues provides for a minimum guaranteed bandwidth and a maximum packet limit (default and maximum: 64 packets). CBWFQ also allows each queue to use excess bandwidth should it be available and not dedicated to other queues. All traffic that is not classified into a class can be classified with flow-based WFQ.

CBWFQ does not address the low delay requirements of video and voice over IP technologies. For these applications, Low-Latency Queuing (LLQ) should be used.

Low Latency Queuing (LLQ)

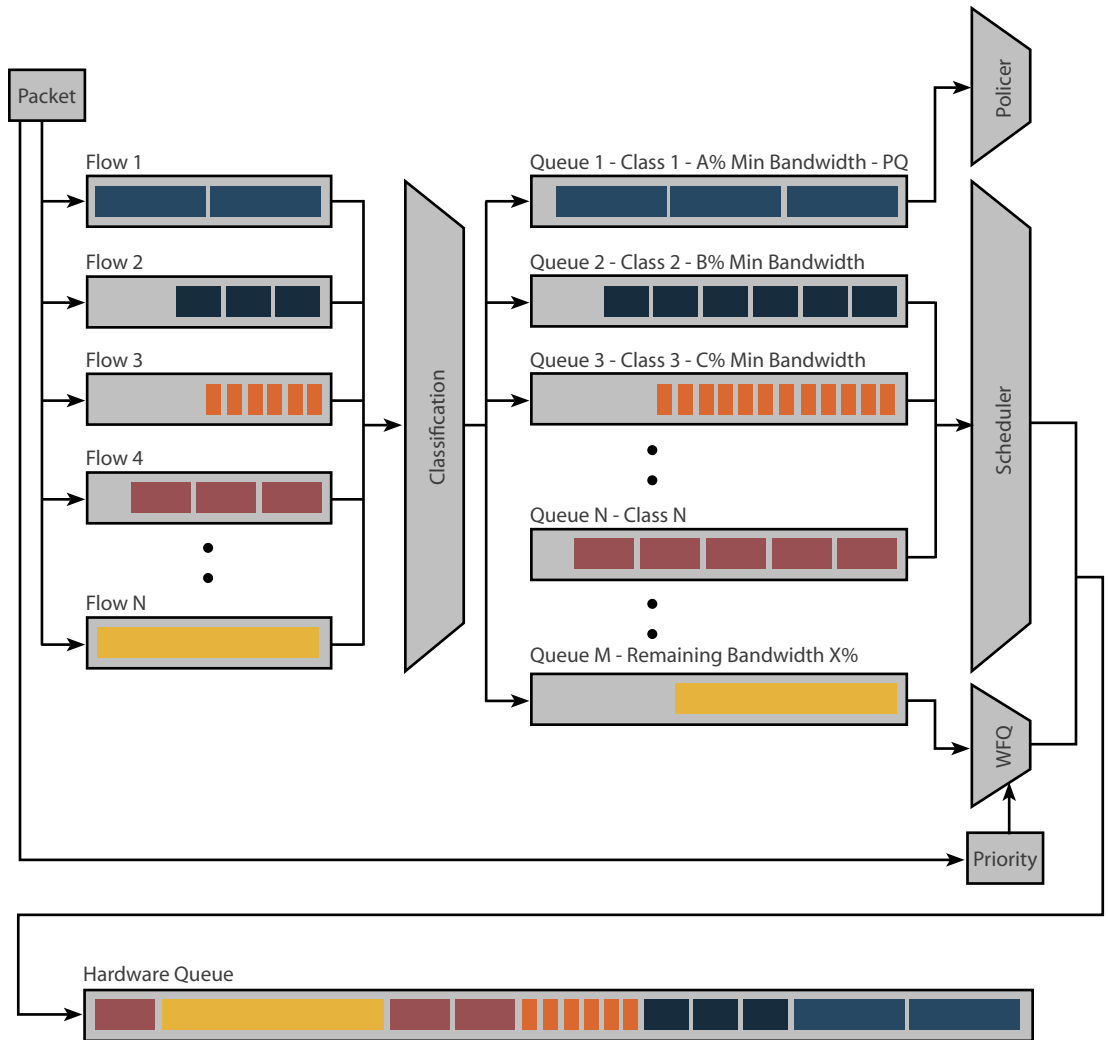


Figure 34 - LLQ Queueing Example

LLQ takes the advantages of CBWFQ and PQ, which is why it can also be called priority queue class-based weighted fair queuing (PQCBWFQ). LLQ essentially does everything the same as CBWFQ but adds the ability to have a priority queue. This priority queue is policed on queue exit when there is congestion; this makes it so that this queue cannot starve any of the other queues. Only the allotted amount of priority bandwidth is guaranteed in congested times.

Control Plane Policing (CoPP)

Control Plane Policing is used to policy the traffic that comes into networking equipment and is destined for the networking equipment itself. With today's concerns over Distributed Denial of Service (DDoS) attacks, Cisco has developed a feature that allows the networking equipment to control excess traffic destined for the control plane. The control plane in this case deals with management traffic and routing traffic, along with other things like keepalive traffic. On Cisco equipment, there are two different types of CoPP supported, aggregate and distributed. Aggregate CoPP provides services for all the Control Plane (CP) traffic that comes in on all line-card interfaces. Distributed CoPP provides services for CP traffic that is received on the interfaces of a line-card. Distributed CoPP services are provided before the aggregate CoPP services. Both aggregate and distributed CoPP services use the MQC policy, class and service maps for configuration.

Software and Hardware Queuing

It is important to understand the differences between software and hardware queuing. All of the queuing mechanisms listed above are maintained in software (Specifically IOS). On top of the software queues which can be used, there is a hardware queue (also called the TX queue or TX Ring) which is associated with each hardware interface. All of the hardware queues operate based on the FIFO queuing technique. This is not configurable. All traffic which comes into a piece of networking equipment goes through a process which dictates how it is handled by the equipment. The first thing that happens before the traffic goes into a software queue is the hardware queue is checked for fullness. If the hardware queue is not full, the traffic is passed directly to the hardware queue. If the queue is full, the traffic progresses into the software queues that are configured.

Congestion Management Configuration

fair-queue

The **fair-queue** [*congestive-discard-threshold* [*dynamic-queues* [*reservable-queues*]]] command is used to enable Weighted Fair Queueing (WFQ). The *congestive-discard-threshold* parameter specifies the number of packets allowed in each queue. By default, this number is 64 but can range from 1 to 4096. The *dynamic-queues* parameter specifies the number of dynamic queues used for best-effort conversations. By default, this number changes based on the bandwidth of the interface. Interface queues by default: < 64kbps = 16 queues, 64kbps ≥ 128kbps = 32 queues, 128kbps ≥ 256kbps = 64 queues, 256kbps ≥ 512kbps = 128 queues, and > 512kbps = 256 queues. The *reservable-queues* parameter is used to specify the number of queues reserved for Resource Reservation Protocol (RSVP). By default, this is 0 but can be a number from 0 to 1000.

Syntax:

```
router(config-if)#fair-queue [congestive-discard-threshold [dynamic-queues [reservable-queues]]]
```

hold-queue

The **hold-queue** *length* **{in | out}** command is used to limit the size of the interface queue. The *length* specifies the size of the queue. By default, the input queue is 75 packets and the output queue is 40 packets, except with asynchronous interfaces whose input and output queues are 10 packets. The *length* can be a number from 0 to 65535. The **in** and **out** parameters specify whether the input or output queue length value is being configured.

Syntax:

```
router(config-if)#hold-queue length {in | out}
```

bandwidth (CBWFQ)

The **bandwidth** *{bandwidth-kbps | remaining percent percentage | percent percentage}* command is used to specify the amount of bandwidth specified to a class inside a policy-map for CBWFQ. The *bandwidth-kbps* parameter specifies the amount of bandwidth to be assigned to the class. The **remaining percent percentage** parameter is used to specify a percentage of bandwidth based on the relative amount of remaining bandwidth. The **percent percentage** parameter is used to specify a percentage of bandwidth based on the absolute percentage.

Syntax:

```
router(config-pmap-c)#bandwidth {bandwidth-kbps | remaining percent percentage | percent percentage}
```

max-reserved-bandwidth

The **max-reserved-bandwidth** *percent* command is used to specify the amount of total interface bandwidth that is allowed to be reserved for RSVP, CBWFQ, LLQ, and IP RTP priority. By default, this is 75% (except for the 7500 series routers where it is 100%). The *percent* parameter is used to specify the percentage allowed to be reserved.

Syntax:

```
router(config-if)#max-reserved-bandwidth percent
```

priority

The **priority** *{bandwidth-kbps | percent percentage} [burst]* command is used to specify the amount of bandwidth to be reserved for a priority queue (LLQ) configured in a class of a policy-map. The *bandwidth-kbps* parameter is used to specify the reserved amount of bandwidth in kbps. The **percent percentage** parameter is used to specify the reserved amount of bandwidth in percentage of available bandwidth. The *burst* parameter is used to specify the burst amount allowed in bytes. By default, the burst is configured to be 200 milliseconds of the configured bandwidth rate.

Syntax:

```
router(config-pmap-c)#priority {bandwidth-kbps | percent percentage} [burst]
```

queue-list queue limit

The **queue-list** *list-number* **queue** *queue-number* **limit** *limit-number* command is used to specify the maximum amount of packets allowed in each of the custom queues. The *list-number* parameter is used to specify the queue list number. This number can be from 1 to 16 and is used to specify order. The *queue-number* parameter is used to specify the queue number. This number can be from 1 to 16. The *limit-number* parameter is used to specify the maximum number of packets which can be queued at the same time. This number can be from 0 to 32767.

Syntax:

```
router(config)#queue-list list-number queue queue-number limit limit-number
```

queue-list queue byte-count

The **queue-list** *list-number* **queue** *queue-number* **byte-count** *byte-count-number* command is used to specify the average amount of bytes forwarded per queue per cycle. The *list-number* parameter is used to specify the queue list number. This number can be from 1 to 16 and is used to specify order. The *queue-number* parameter is used to specify the queue number. This number can be from 1 to 16. The *byte-count-number* parameter is used to specify the average number of bytes which are allowed to be forwarded per queue per cycle.

Syntax:

```
router(config)#queue-list list-number queue queue-number byte-count byte-count-number
```

queue-list protocol

The **queue-list** *list-number* **protocol** *protocol-name* *queue-number* *queue-keyword* *keyword-value* command is used to establish queueing priority based upon protocol. The *list-number* parameter is used to specify the queue list number. This number can be from 1 to 16 and is used to specify order. The *protocol-name* parameter is used to specify the protocol. The *queue-number* parameter is used to specify the queue number. This number can be from 1 to 16. The *queue-keyword* parameter is used to specify the matching keyword; possible options include **fragments**, **gt**, **list**, **lt**, **tcp**, and **udp**. The *keyword-value* parameter is used in conjunction with the *queue-keyword* parameter to specify matching criteria.

Syntax:

```
router(config)#queue-list list-number protocol protocol-name queue-number queue-keyword keyword-value
```

queue-list interface

The **queue-list** *list-number* **interface** *interface-type* *interface-number* *queue-number* command is used to establish queueing priority based upon interface. The *list-number* parameter is used to specify the queue list number. This number can be from 1 to 16 and is used to specify order. The *interface-type* *interface-number* parameters are used to specify an interface. The *queue-number* parameter is used to specify the queue number. This number can be from 1 to 16.

Syntax:

```
router(config)# queue-list list-number interface interface-type interface-number
queue-number
```

queue-list default

The **queue-list** *list-number* **default** *queue-number* command is used to specify a default queue which be used for traffic that does not match any other commands. The *list-number* parameter is used to specify the queue list number. This number can be from 1 to 16 and is used to specify order. The *queue-number* parameter is used to specify the queue number. This number can be from 1 to 16.

Syntax:

```
router(config)#queue-list list-number default queue-number
```

Domain 6 - Congestion Avoidance Methods

Tail-Drop

Tail Drop is the default behavior for queues on Cisco equipment. *Tail drop* works by simply dropping the packets which are destined for an already full queue. The main problem with this is that no priority is given to any packet dropped, so high priority traffic, which should not be dropped, is and low priority traffic, which should be dropped, is not.

Weighted Random Early Detection (WRED)

WRED is different from all the queue management options because it is geared to congestion avoidance not management. WRED does this by selectively dropping packets based on the average queue size and the priority of the packet. WRED is configured with a minimum queue threshold, a maximum queue threshold and a mark probability denominator. The minimum and maximum thresholds are used by WRED to determine when to start dropping packets and when to drop all packets. Between the two thresholds, packets are dropped linearly up to the maximum threshold. When the average queue size is at the maximum threshold, WRED drops the most amount of packets based on the mark probability denominator. The mark probability denominator is the fraction of packets dropped while at the maximum threshold. By default it is set to 10. WRED calculates the average queue size based on the following formula:

$$\text{Average} = (\text{old_average} \times (1 - 2^{-n})) + (\text{current_queue_size} \times 2^{-n})$$

By default, *n* (exponential-weighting-constant) = 9. The lower the number, the quicker WRED reacts to queue fluctuations; the higher the number, the slower it reacts. Cisco recommends not changing this default.

WRED is intended mainly to be used to control TCP traffic as UDP traffic does not require a connection or flow control, which is influenced by packet loss. TCP has a built-in congestion control mechanism. This mechanism is called *windowing* and is used to determine the number of TCP packets sent before an acknowledgement. When a *window* is set to 1, for every packet sent a packet is required to be sent back to the sender. This can quickly become inefficient. Because of this, TCP allows the *window* to be enlarged as TCP traffic is sent and acknowledged successfully. However, in congested times, some traffic can be dropped unexpectedly which causes TCP to retransmit and to reset the *window* to 1. When this happens with all the TCP flows, it is called *TCP global synchronization*. This has the effect of causing all traffic to slow. Without WRED, flows tend to have high peaks and valleys. As congestion occurs, global synchronization commences and all traffic slows. Then, as the congestion goes away, all flows ramp traffic back up until congestion happens again, and global synchronization occurs again. WRED limits the amount of time that global synchronization occurs and by doing this makes the peaks and valleys of the traffic lower, which raises the average amount of bandwidth.

WRED is also IP precedence, DSCP, and RSVP aware as it has the ability to select which packets are selectively dropped and of these, WRED can also be configured to have different configuration parameters for each precedence or DSCP setting.

The following table lists the Cisco default profiles for IP Precedence and DSCP:

IP Precedence	Minimum Threshold	Maximum Threshold	Mark Probability Denominator	Calculated Maximum Percent Discarded
0	20	40	10	10% (1/10)
1	22	40	10	10% (1/10)
2	24	40	10	10% (1/10)
3	26	40	10	10% (1/10)
4	28	40	10	10% (1/10)
5	31	40	10	10% (1/10)
6	33	40	10	10% (1/10)
7	35	40	10	10% (1/10)
RSVP	37	40	10	10% (1/10)

Table 8 - IP Precedence WRED Profile

DSCP	Minimum Threshold	Maximum Threshold	Mark Probability Denominator	Calculated Maximum Percent Discarded
AF11, AF21, AF31, AF41	33	40	10	10% (1/10)
AF12, AF22, AF32, AF42	28	40	10	10% (1/10)
AF13, AF23, AF33, AF43	24	40	10	10% (1/10)
EF	37	40	10	10% (1/10)

Table 9 - DSCP WRED Profile

WRED and Queuing

How WRED is implemented depends upon the configuration. WRED does not work in conjunction with WFQ, CQ or PQ. This means that primarily it is used with CBWFQ and can be used with LLQ. By default, tail drop is used on all queue management options. WRED can also be configured to work directly on an interface. However, for this to work, the FIFO queuing mechanism must be used. When configuring WRED for use with CBWFQ and LLQ, the configuration commands are inserted under the **class** configuration mode. WRED works in both CBWFQ and LLQ the same way. It can be independently configured per traffic class. As with CBWFQ and LLQ, the individual classes each have an independent FIFO queue. WRED makes the decision to drop or not before traffic goes into this queue. WRED is also limited by one other thing. When configuring it with LLQ, it cannot be used in conjunction with the **priority** command. This means that WRED is not used in conjunction with the priority queue, but will work on all other classes created under LLQ. Configuring WRED on all other classes which contain TCP traffic is the recommended implementation.

The following shows WRED configured on both a physical interface and configured within CBWFQ:

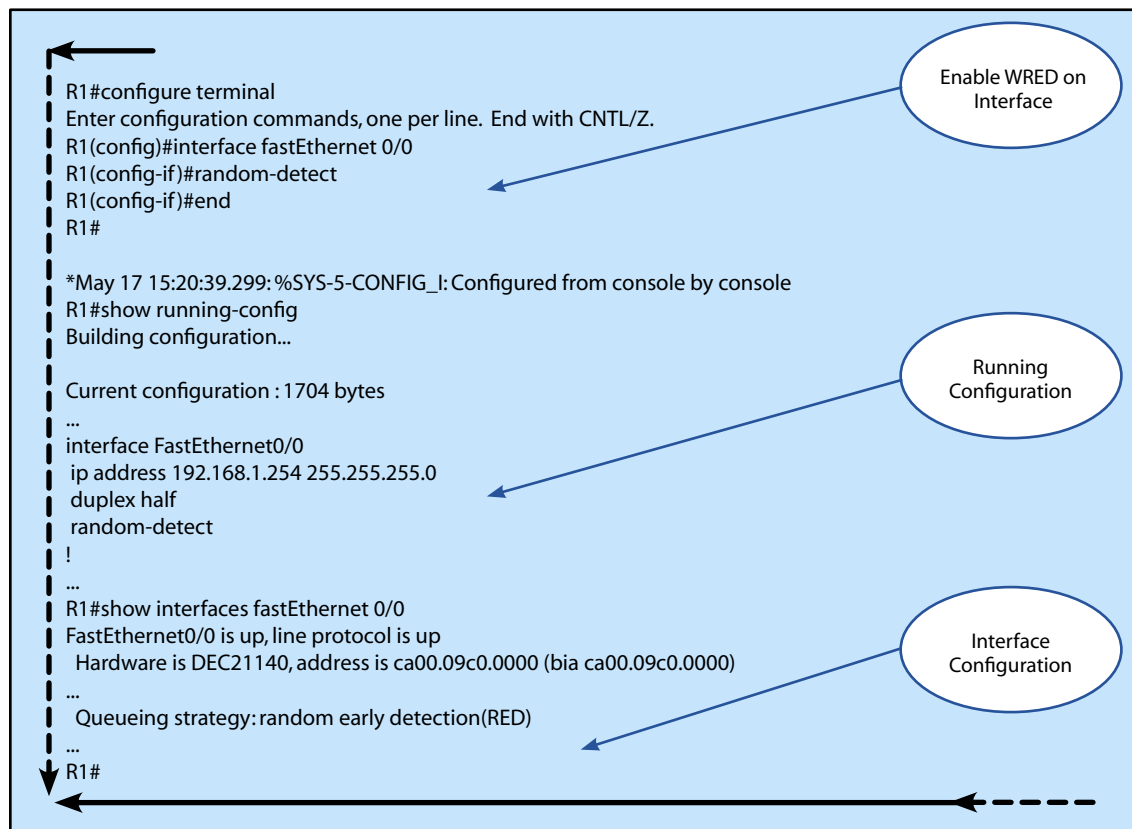


Figure 35 - WRED Interface Example

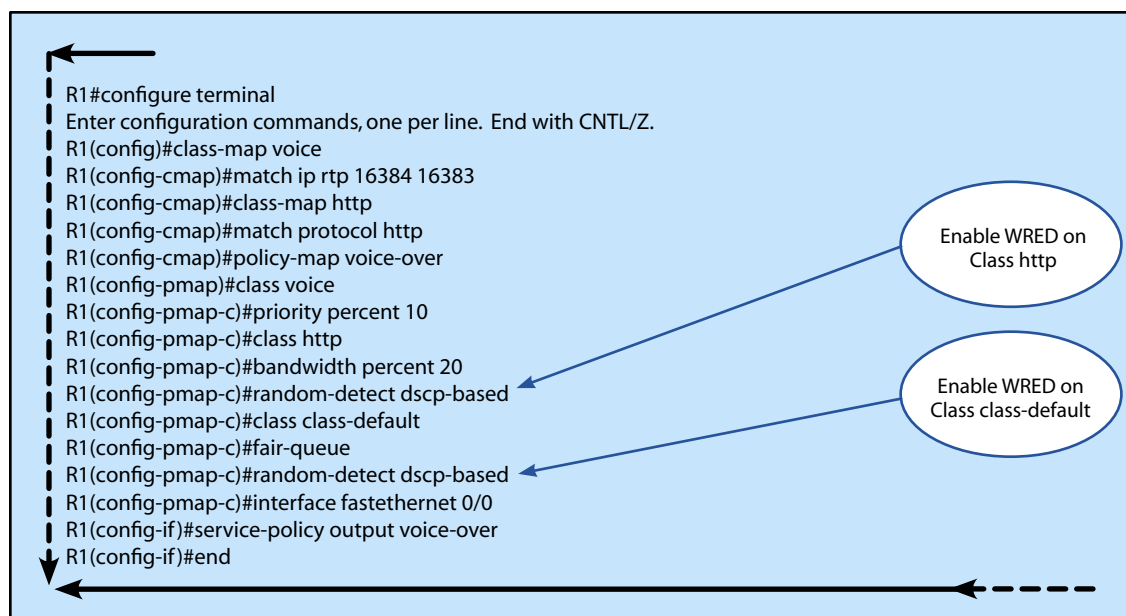


Figure 36 – CBWFQ/LLQ WRED Example

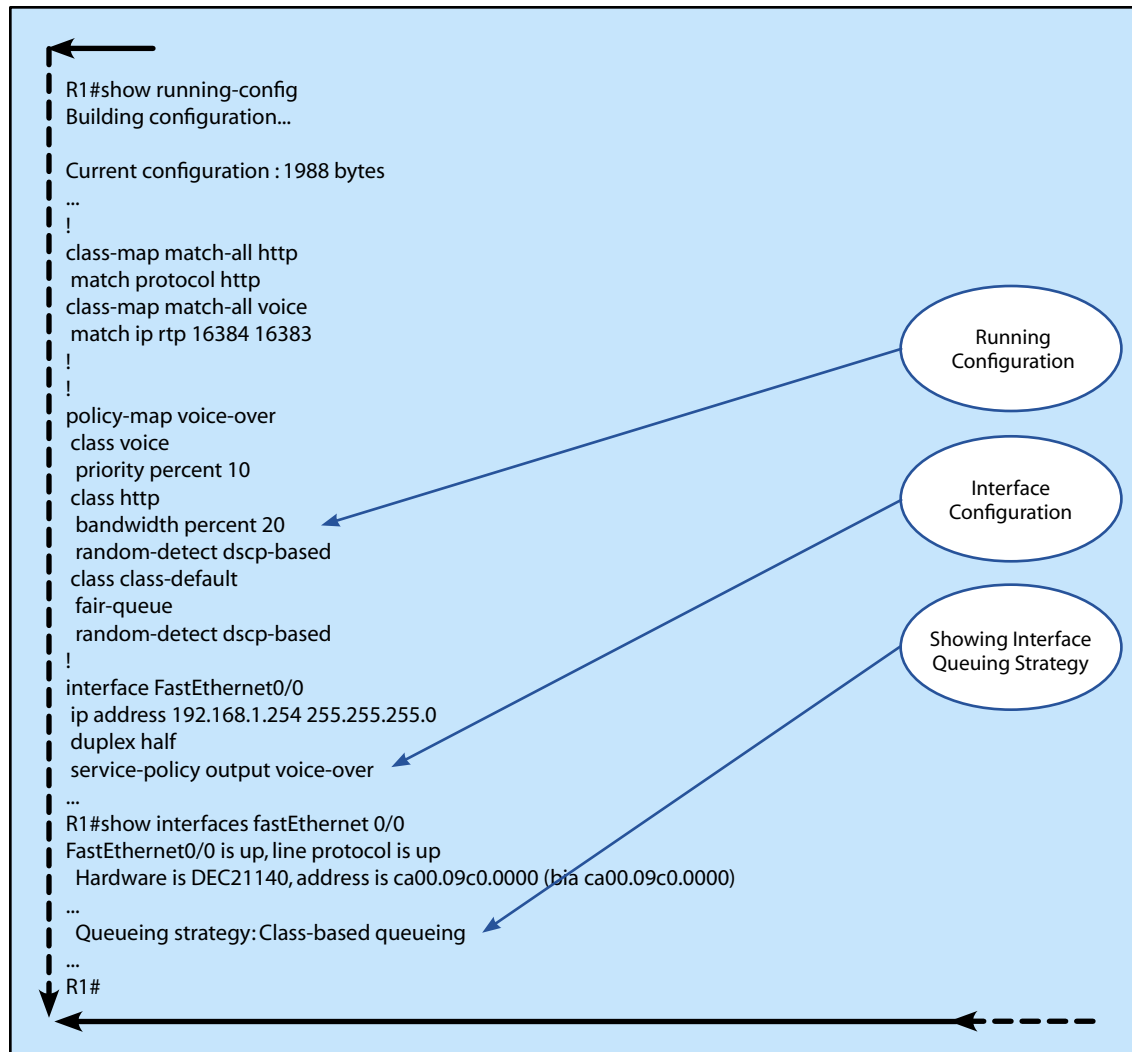


Figure 37 – CBWFQ/LLQ WRED Running-Config

Explicit Congestion Notification (ECN) and WRED

ECN is a field which can be used in conjunction with DSCP. It is used to alert of congestion on the network. The specific ECN values are listed in table 5. When congestion is indicated on network traffic, the endpoints lower the TCP window settings. This action slows the traffic and reduces congestion. All clients do not support ECN; clients which do not support ECN marking will ignore it.

WRED works with the ECN field. When configured to support ECN, WRED changes how it treats traffic which is between the minimum and maximum thresholds. Ordinarily, when the average queue size is between the minimum and maximum threshold, packets are dropped based on the mark probability denominator and the average size of the queue. When ECN is enabled, this traffic, instead of being dropped, is marked with an ECN value which notates congestion ('11'). This is then used by the TCP endpoints to throttle traffic by lowering the TCP *window*. If the traffic indicates that neither endpoint supports ECN ('00')

then the traffic will be dropped. If the average queue size at any time goes above the maximum threshold then traffic is dropped based on the mark probability denominator. The use of ECN is preferred on many networks because no traffic is dropped while under the maximum threshold.

WRED support with ECN is configured through the use of the **random-detect ecn** command.

Configuration Avoidance Configuration

random-detect

The **random-detect [dscp-based | prec-based]** command is used to enable WRED on an interface or class of a policy-map. The **dscp-based** parameter is used to specify the use of DSCP values. The **prec-based** parameter is used to specify the use of IP precedence values.

Syntax:

```
router(config-if)#random-detect [dscp-based | prec-based]
```

or

```
router(config-pmap-c)#random-detect [dscp-based | prec-based]
```

random-detect dscp

The **random-detect dscp dscp-value min-threshold max-threshold [max-probability-denominator]** command is used to set the minimum and maximum thresholds for DSCP values. The *dscp-value* parameter specifies the DSCP value to be configured. This can be a number from 0 to 63 or the keywords **af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, cs1, cs2, cs3, cs4, cs5, cs7, ef, or rsvp**. The *min-threshold* parameter specifies the minimum threshold for average queue length to start to be dropped in number of packets. This can be from 1 to 4096. The *max-threshold* specifies the maximum threshold for average queue length to be completely dropped in number of packets. This can be from 1 to 4096. The *max-probability-denominator* parameter is used to specify the probability denominator. This is used to show the amount of packets which are dropped when at maximum threshold (but not over). The denominator is used as a fraction of packets. The default is 10 on most Cisco equipment, which means one packet in ten will be dropped when at the maximum threshold.

Syntax:

```
router(config-if)#random-detect dscp dscp-value min-threshold max-threshold [max-probability-denominator]
```

or

```
router(config-pmap-c)#random-detect dscp dscp-value min-threshold max-threshold [max-probability-denominator]
```

random-detect precedence

The **random-detect precedence** *{precedence | rsvp} min-threshold max-threshold max-probability-denominator* command is used to set the minimum and maximum thresholds for IP precedence values. The *precedence* parameter specifies the IP precedence value to be configured. This can be a number from 0 to 7. The **rsvp** parameter is used to specify RSVP traffic. The *min-threshold* parameter specifies the minimum threshold for average queue length to start to be dropped in number of packets. This can be from 1 to 4096. The *max-threshold* specifies the maximum threshold for average queue length to be completely dropped in number of packets. This can be from 1 to 4096. The *max-probability-denominator* parameter is used to specify the probability denominator. This is used to show the amount of packets which are dropped when at maximum threshold (but not over). The denominator is used as a fraction of packets. The default is 10 on most Cisco equipment, which means one packet in ten will be dropped when at the maximum threshold.

Syntax:

```
router(config-if)#random-detect precedence {precedence | rsvp} min-threshold max-threshold max-probability-denominator
```

or

```
router(config-pmap-c)#random-detect precedence {precedence | rsvp} min-threshold max-threshold max-probability-denominator
```

random-detect ecn

The **random-detect ecn** command is used to enable ECN support within WRED.

Syntax:

```
router(config-pmap-c)#random-detect ecn
```

random-detect exponential-weighting-constant

The **random-detect exponential-weighting-constant** *exponent* command is used to specify the exponential weight factor used in average queue calculations. The *exponent* parameter is used to specify the exponent to be used. By default, this value is 9 but can be from 1 to 16.

Syntax:

```
router(config-if)#random-detect exponential-weighting-constant exponent
```

or

```
router(config-pmap-c)#random-detect exponential-weighting-constant exponent
```

Domain 7 - Traffic Policing and Shaping

Token Bucket Concept

Both traffic policing and shaping have one thing in common and that is the use of a token bucket for measuring traffic conformance. A token bucket is used to represent the amount of traffic that is allowed to be sent through an interface. This is done through the adding and subtracting of tokens into a 'bucket'. Generally the single token bucket concept involves a traffic mean rate or Committed Information Rate (CIR), a burst size or Committed Burst (Bc) which is represented in bits for shaping and bytes for policing, and a measurement interval (Tc). There is a relationship between these three values: $CIR = Bc/Tc$. At every measurement interval, an amount of tokens which represents the CIR is added to the 'bucket' and when traffic needs to be transmitted it is subtracted from the 'bucket'. The Bc is used to represent the size of the 'bucket'. If there is sufficient Bc (excess capacity in the 'bucket') to allow a traffic burst the traffic is sent. If there is not capacity in the bucket the option is to drop the traffic which exceeds the bucket size, mark the traffic or to permit the traffic without marking.

This concept is extended when dealing with a dual rate policer (or dual token). With a dual rate policer, the initial bucket works as described above; however, a second bucket also exists. This second bucket is based on a traffic peak rate or Peak Information Rate (PIR), a burst size or Burst Excess (Be) and a measurement interval (Tp). There is a relationship between these three values: $PIR = Be/Tp$. This second bucket offers a third option for conformance, a violate action. This violate action occurs should the traffic not only exceed the Bc bucket size but also the Be bucket size. This means, in more general language, that the traffic exceeded both the CIR and the PIR.

Traffic Policing Basics

Traffic policing handles excesses in traffic rate and burst either by completely dropping the excess traffic or remarking it with lower priority depending on the situation. Cisco handles policing through Committed Access Rate (CAR). CAR uses the newer MQC configuration options with class, policy and service maps which allow for easy configuration. CAR must first classify the traffic that is intended to be policed. This is done through the use of the **policy-map** and **class-map** commands. CAR has the ability to classify this traffic using the incoming interface, all IP traffic, the IP Precedence, MAC address, MPLS EXP field or an IP access list. CAR must then be configured with the rate limiting parameters. This is done through the **police** command. With CAR, the CIR, the normal burst and the excess burst are able to be set. The normal burst amount is used to set the depth of the 'bucket', the excess burst provides a capability for excessive burst. This excess burst is implemented by allowing the traffic to borrow an amount of tokens to service the traffic.

CAR does not provide for any traffic shaping. It simply provides three actions should the traffic exceed or violate conformance. These three actions are either to drop the traffic, remark the traffic with a lower priority (IP precedence or DSCP), or to ignore the violation and send the packet.

$$\begin{aligned} \text{NormalBurst} &= \text{CIR} \times 1 \text{ Byte} / 8 \text{ Bits} \times 1.5 \\ \text{ExtendedBurst} &= 2 \times \text{NormalBurst} \end{aligned}$$

Figure 38 - Cisco Recommended Values

Traffic Policing Configuration

Basics

In order to configure single and dual rate policing as well as percentage based policing, the same **police** command is used. The syntax differs in minor ways depending on which type of policing is needed. For details of each version of the **police** command see below. Other than the minor differences in command syntax, all three are configured in the same way through using the MQC syntax as detailed earlier in this manual. The following shows an example of policing. The **police** command in this example can be exchanged for any of the versions of the command and work correctly.

```

R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#class-map police
R1(config-cmap)#match ip dscp default
R1(config-cmap)#policy-map normal-police-policy
R1(config-pmap)#class police
R1(config-pmap-c)#police 512000 4096 4096 conform-action transmit exceed-action drop
R1(config-pmap-c-police)#exit
R1(config-pmap)#interface fastethernet 0/0
R1(config-if)#service-policy input normal-police-policy
R1(config-if)#end

```

Figure 39 - Single-Rate Policing Example

```

R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#class-map police
R1(config-cmap)#match ip dscp default
R1(config-cmap)#policy-map normal-police-policy
R1(config-pmap)#class police
R1(config-pmap-c)#police cir 512000 pir 1544000 conform-action transmit exceed-action set-frde-
transmit violate-action drop
R1(config-if)#interface s1/0.1
R1(config-subif)#service-policy output normal-police-policy
R1(config-subif)#end

```

Figure 40 - Dual-Rate Policing Example

If configuration with CBWFQ is wanted then the structure of the MQC commands can accommodate. As seen in the example below, MQC syntax allows the nesting of a policy maps. This allows the person configuring to have the flexibility to use shaping, policing and CBWFQ together at different class levels. An example of this is also below.

```

R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#policy-map child
R1(config-pmap)#class class-default
R1(config-pmap-c)#bandwidth percent 75 (CBWFQ Config)
R1(config-pmap-c)#exit
R1(config-pmap)#exit
R1(config)#policy-map parent
R1(config-pmap)#class class-default
R1(config-pmap-c)#police 64000 8000 8000 conform-action transmit exceed-action drop (Policing Config)
R1(config-pmap-c)#service-policy child
R1(config-pmap-c)#end
R1#

```

Figure 41 - MQC Policing, Shaping and CBWFQ Nesting Example

police

Single-Rate

The **police** *bps* [*burst-normal*] [*burst-max*] **conform-action** *action* **exceed-action** *action* [**violate-action** *action*] command is used to configure traffic policing in policy-map class configuration mode. The *bps* parameter is used to specify the average traffic rate in bits per second. This value can be set from 8000 to 200000000. The *burst-normal* parameter is used to specify the normal burst size in bytes. This value can be set from 1000 to 51200000. By default, this value is 1500. The *burst-max* parameter is used to specify the maximum burst size in bytes. This value can be set from 1000 to 51200000. The **conform-action** *action* parameter is used to specify the action that will be taken should the traffic conform to the policy. The **exceed-action** *action* parameter is used to specify the action that will be taken should the traffic exceed the policy. The **violate-action** *action* parameter is used to specify the action that will be taken should the traffic violate the policy.

Dual-Rate

The **police** [*cir cir*] [**bc** *conform-burst*] [**pir** *pir*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action*] [**violate-action** *action*]]] command is used to configure dual-rate traffic policing in policy-map class configuration mode. The *cir cir* parameter is used to specify the committed traffic rate in bits per second. This value can be set from 8000 to 200000000. The **bc** *conform-burst* parameter is used to specify the conform burst size in bytes. This value can be set from 1000 to 51200000. The **pir** *pir* parameter is used to specify the peak traffic rate in bits per second. This value can be from 8000 to 200000000. The **be** *peak-burst* parameter is used to specify the peak burst size in bytes. This value varies by platform and interface. The **conform-action** *action* parameter is used to specify the action that will be taken should the traffic conform to the policy. The **exceed-action** *action* parameter is used to specify the action that will be taken should the traffic exceed the policy. The **violate-action** *action* parameter is used to specify the action that will be taken should the traffic violate the policy.

Percent

The **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*]] command is used to configure traffic policing based on the percentage of interface bandwidth in policy-map class configuration mode. If the **police** command is used in a child policy-map, the bandwidth used for percent calculation comes from the parent policy-map. If a bandwidth is not specified in the parent policy-map, the interface bandwidth would again be the measure. The **cir percent** *percent* parameter is used to specify the committed traffic rate. The **bc conform-burst-in-msec** parameter is used to specify the conform burst size in msec. By default, this value is 4 msec. The **pir percent** *percent* parameter is used to specify the peak traffic rate. The **be peak-burst-in-msec** parameter is used to specify the peak burst size in msec. By default, this value is 4 msec. The **conform-action** *action* parameter is used to specify the action that will be taken should the traffic conform to the policy. The **exceed-action** *action* parameter is used to specify the action that will be taken should the traffic exceed the policy. The **violate-action** *action* parameter is used to specify the action that will be taken should the traffic violate the policy.

drop	Drops the packet.
set-clp-transmit <i>value</i>	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 <i>value</i>	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 <i>value</i>	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 <i>value</i>	Sets the IP differentiated services code point (DSCP) value and transmits the packet with the new IP DSCP value
set-dscp-tunnel-transmit <i>value</i>	Sets the DSCP value (0 to 63) in the tunnel header of a Layer 2 Tunnel Protocol Version 3 (L2TPv3) tunneled packet for tunnel marking and transmits the packet with the new value
set-frde-transmit <i>value</i>	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 <i>value</i>	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 <i>value</i>	Sets the MPLS EXP field value in the topmost MPLS label header at the input and/or output interfaces
set-prec-transmit <i>value</i>	Sets the IP precedence and transmits the packet with the new IP precedence value

table continued on next page

set-prec-tunnel-transmit <i>value</i>	Sets the precedence value (0 to 7) in the tunnel header of an L2TPv3 tunneled packet for tunnel marking and transmits the packet with the new value
set-qos-transmit <i>value</i>	Sets the qos-group value and transmits the packet with the new qos-group value
transmit	Transmits the packet. The packet is not altered

Table 10 - action options

Syntax:

```
router(config-pmap-c)#police bps [burst-normal] [burst-max] conform-action action
exceed-action action [violate-action action]
```

or

```
router(config-pmap-c)#police [cir cir] [bc conform-burst] [pir pir] [be peak-burst] [con-
form-action action [exceed-action action [violate-action action]]]
```

or

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


Traffic Shaping

Basics

Traffic shaping handles excesses in traffic rate and burst differently from a traffic policer. The traffic shaper uses a token bucket but differs from a policer in that it has an additional queue. This queue is used by the shaper by placing the traffic that exceeds the CIR into it. The queue will be emptied as the traffic slows allowing for a smoother traffic profile.

On Cisco equipment there are three different implementations of traffic shaping: Class-based traffic shaping, Generic Traffic Shaping (GTS) and Frame Relay Traffic Shaping (FRTS). These three different mechanisms perform the shaping in a very similar way but the configuration and types of queues used differ between them.

Type	Class-Based	GTS	FRTS
CLI	Applied per Class	Applied per Interface or Subinterface Supports traffic groups (traffic group command)	Applies to all Virtual Circuits (VC) on an interface.
Queues Supported	Supports WFQ and Class-Based WFQ (CBWFQ)	Supports WFQ	Supports WFQ, WFQ with priority, CQ, PQ, and FIFO per VC.

Figure 42 - Traffic Shaping Mechanism Differences

Class-Based traffic shaping is the Cisco-recommended traffic shaper mainly because it allows the most flexibility of all the shapers. Class-Based traffic shaping by default uses WFQ but can use CBWFQ if hierarchical configuration is used. Class-Based traffic shaping allows different classes to be created for each different type of traffic. As stated above, it also has the ability to be hierarchical, meaning it allows the traffic to be shaped initially. If the shaped traffic still exceeds a threshold, the traffic is then passed to CBWFQ queuing or to be policed. Class-Based traffic shaping is based on the MQC policy, class and service-maps which allow for easy CLI configuration. Class-Based traffic shaping is limited to shaping outbound traffic.

Generic Traffic Shaping (GTS) is different from class-based traffic shaping as it is applied not by class, but by interface and/or by access-list. It can also be used on almost all types of interface and encapsulation, with the exception of MLP interfaces and ISDN, dialer and GRE tunneling interfaces on 7500 series routers. With GTS, the shaping is configured through the **traffic-shape** command. GTS also has the capability to adapt the shaping based on the FECN and BECN on frame relay interfaces.

Frame Relay Traffic Shaping (FRTS) is different from the other traffic shapers because it is applied on a per-VC basis. This mapping to a VC is done through the use of the **map-class** command. Parameters of traffic shaping are configured on the map-class and the VCs are associated with the map class. Cisco equipment also supports Foresight on some of their switches which can be enabled on the map-class by using the **frame-relay adaptive-shaping foresight** command.

Shaping Adaptation

Cisco traffic shaping also has the ability to modify the shaping behavior based on a number of criteria. These criteria include Frame Relay FECN and BECN fields, Cisco Foresight and based on high priority voice traffic which is in the LLQ priority queue. When using the BECN Frame-Relay field, the networking equipment monitors the BECN field on incoming traffic. Should the field be set on incoming traffic, the equipment will lower the traffic rate by $\frac{1}{4}$ until the lower bound of the CIR. Once the BECN field is not set, the rate then is increased by $\frac{1}{16}$ until the CIR is reached. All of this is done with the **shape adaptive** command. The FECN Frame-Relay field is used by translating it into a BECN bit then uses this information to rate the traffic as above. This can only be done in the *class-default* class and is done with the **shape fecn-adapt** command. Traffic shaping can also be setup to monitor the traffic in the LLQ priority queue which typically contains voice traffic. This type of adaptation works by monitoring the LLQ Priority queue. Should there be traffic in this queue, the traffic rate on the *class-default* class lowers to the minCIR rate (configured with the **shape adaptive** command). Once traffic ceases to exist in this queue, the traffic rate is raised to the CIR.

Another type of adaptation is supported with FRTS; this type of adaptation relies on the Cisco proprietary Foresight congestion protocol which runs on Cisco WAN switching equipment. Foresight works by having the networking equipment talk to each other and let each other know the state of the network. Should there be congestion, the networking equipment will alert other equipment to the congestion. Based on this information, the **frame-relay adaptive-shaping foresight** command is used to pay attention and alter the rate.

Traffic Shaping Configuration

Traffic shaping configuration is very similar to policing configuration. All the same MQC commands are used, but the only difference is the use of the **shape** command instead of the **police** command.

```

R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#class-map shape
R1(config-cmap)#match ip dscp default
R1(config-cmap)#policy-map normal-shape-policy
R1(config-pmap)#class shape
R1(config-pmap-c)#shape average 512000
R1(config-pmap-c)#interface s1/0.1
R1(config-subif)#service-policy output normal-shape-policy
R1(config-subif)#end
R1#

```

Figure 43 - Shaping Example

shape

The **shape** **{average | peak}** *mean-rate* [*burst-size*] [*excess-burst-size*] command is used to configure traffic shaping. The **average** parameter specifies the use of average rate shaping. The **peak** parameter specifies the use of peak rate shaping. The *mean-rate* parameter specifies the committed information rate (CIR) in bits per second. The *burst-size* parameter specifies the committed burst size (Bc) in bits per second. The *excess-burst-size* specifies the excess burst size (Be) in bits per second.

Syntax:

```
router(config-pmap-c)#shape {average | peak} mean-rate [burst-size] [excess-burst-size]
```

shape percent

The **shape** **{average | peak} percent** *percentage* [*sustained-burst-in-msec ms*] [**be** *excess-burst-in-msec ms*] [**bc** *committed-burst-in-msec ms*] command is used to configure traffic shaping based on a percentage of interface bandwidth. The **average** parameter specifies the use of average rate shaping. The **peak** parameter specifies the use of peak rate shaping. The **percent** *percentage* parameter is used to specify the average or peak shaping bandwidth based on available interface bandwidth. The *sustained-burst-in-msec ms* parameter is used to specify the sustained burst amount allowed by the first token bucket. The **be** *excess-burst-in-msec ms* parameter is used to specify the excessive burst allowed by the second token bucket. The **bc** *committed-burst-in-msec ms* parameter is used to specify the committed burst allowed by the second token bucket.

Syntax:

```
router(config-pmap-c)# shape {average | peak} percent percentage [sustained-burst-in-msec ms] [be excess-burst-in-msec ms] [bc committed-burst-in-msec ms]
```

shape fecn-adapt

The **shape fecn-adapt** command is used to enable the translation of FECN field into a BECN message. This information is then used to adapt traffic based on the BECN field status.

Syntax:

```
router(config-pmap-c)#shape fecn-adapt
```

shape fr-voice-adapt

The **shape fr-voice-adapt** [**deactivation** *seconds*] command is used to enable adaption based on the status of the LLQ priority queue. The **deactivation** *seconds* parameter is used to set the number of seconds that must elapse before the traffic rate is raised after the LLQ priority queue empties.

Syntax:

```
router(config-pmap-c)#shape fr-voice-adapt [deactivation seconds]
```

Troubleshooting

There are a number of commands which can be used to troubleshoot traffic policing and shaping. Most of these commands are reviewed in Domain 3.

Domain 8 - Link Efficiency Mechanisms

Link and Fragmentation and Interleaving (LFI)

Multilink PPP (MLP)

MLP provides the capability to fragment and interleave packets together. This allows for time-sensitive packets to be forwarded out of the hardware queue (FIFO) faster when large datagram packets can potentially slow these time-sensitive packets. It does this by splitting larger packets into fragments, which can then be interleaved with these time sensitive packets; this is called Link Fragmentation and Interleaving (LFI). MLP is configured using the **ppp multilink fragment-delay** and **ppp multilink interleave** commands.

Fragment Size Formulation

The fragment size which is used with MLP is found using a simple formula.

$$\text{FragmentSize} = \text{MaxDelay} \times \text{Bandwidth}$$

Figure 44 - Fragment Formula

In order to be clear about how fragment size is figured, a few examples are in order. If the bandwidth of an interface is 64,000 bps and the fragment-delay is 10ms, the fragment size is 640 bits or 80 bytes. This is found as follows:

$$\begin{aligned} 64,000 \times .01(10\text{ms}) &= 640\text{bits} \\ 640 / 8 &= 80\text{Bytes} \end{aligned}$$

Figure 45 - Fragment Size Example

If the bandwidth of an interface is 512,000 bps and a fragment-delay is 30ms, the fragment size is 15360 bits or 1920 bytes. This is found as follows:

$$\begin{aligned} 512,000 \times .03(30\text{ms}) &= 15,360\text{bits} \\ 15,360 / 8 &= 1920\text{Bytes} \end{aligned}$$

Figure 46 - Fragment Size Example

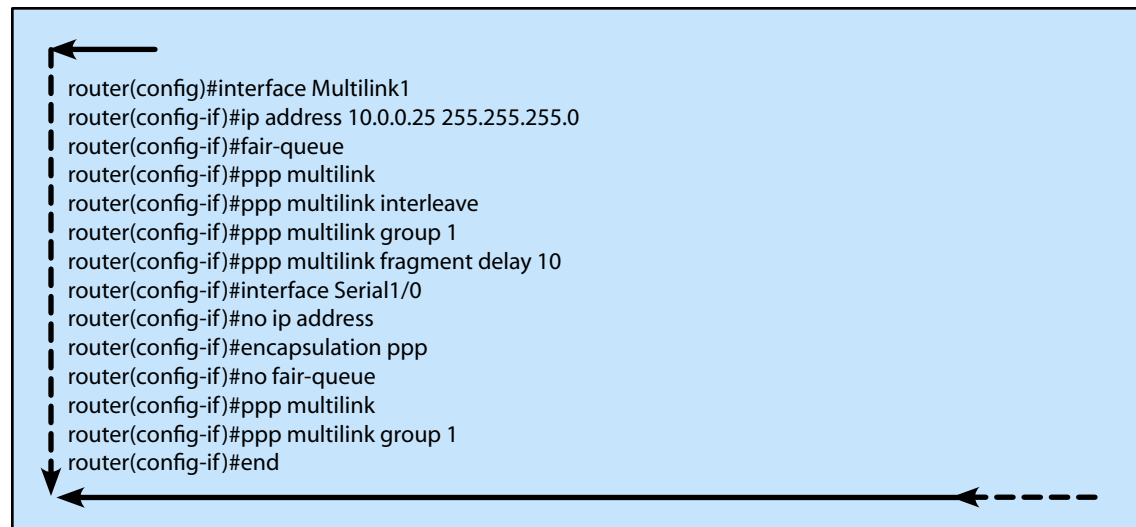
Frame Relay LFI (FRF.12)

LFI also exists for Frame Relay interfaces and is mainly used through the standard Frame-Relay Forum .12 (FRF.12). The function of the LMI with FRF.12 is the same as with MLP. However, there are some differences. With FRF.12, FRTS must be used and is enabled through the **frame-relay fragment** command.

LFI Configuration

MLP Configuration

Multilink PPP is able to be configured within the multilink interface configuration. An example of this configuration is shown here:



```
router(config)#interface Multilink1
router(config-if)#ip address 10.0.0.25 255.255.255.0
router(config-if)#fair-queue
router(config-if)#ppp multilink
router(config-if)#ppp multilink interleave
router(config-if)#ppp multilink group 1
router(config-if)#ppp multilink fragment delay 10
router(config-if)#interface Serial1/0
router(config-if)#no ip address
router(config-if)#encapsulation ppp
router(config-if)#no fair-queue
router(config-if)#ppp multilink
router(config-if)#ppp multilink group 1
router(config-if)#end
```

Figure 47 - LFI-MLP Example

FRF.12 Configuration

The configuration of FRF.12 is a little different than MLP. For FRF.12, the configuration is done within a Frame-Relay map-class. An example of this configuration is shown here:

```
router#configure terminal
router(config)#interface Serial1/0
router(config-if)#no ip address
router(config-if)#encapsulation frame-relay
router(config-if)#serial restart-delay 0
router(config-if)#frame-relay traffic-shaping
router(config-if)#interface Serial1/0.1 point-to-point
router(config-if)#ip address 10.0.0.6 255.255.255.252
router(config-if)#frame-relay class example
router(config-if)#frame-relay interface-dlci 301
router(config-if)#map-class frame-relay example
router(config-map-class)#frame-relay fragment 100
router(config-map-class)#frame-relay traffic-rate 64000 128000
router(config-map-class)#frame-relay adaptive-shaping becn
router(config-map-class)#frame-relay fair-queue
router(config-map-class)#end
```

Figure 48 - LFI - FRF.12 Example

FRF.12 Troubleshooting

This command is used to display Frame-Relay fragment information. The following highlights the most important parts.

```
router#show frame-relay fragment
interface      dlci frag-type size in-frag out-frag dropped-frag
Se1/0.1       301 end-to-end 100 0    48    0
router#
```

Figure 49 - show frame-relay fragment

This command is used to display information about Frame-Relay PVC configuration and statistics. The following highlights the most important parts.

```

router#show frame-relay pvc
PVC Statistics for interface Serial1/0 (Frame Relay DTE)

      Active  Inactive  Deleted  Static
Local      3         0         0         0
Switched   0         0         0         0
Unused     0         0         0         0

DLCI = 301, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0.1

input pkts 820      output pkts 820      in bytes 68307
out bytes 68550    dropped pkts 0      in pkts dropped 0
out pkts dropped 0  out bytes dropped 0
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 770  out bcast bytes 63986
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
Shaping adapts to BECN
pvc create time 00:55:51, last time pvc status changed 00:54:12

```

Figure 50 - show frame-relay pvc

Compression

Header Compression

Header compression on Cisco equipment has two forms: TCP header compression and RTP header compression. Both provide for compression which can provide for a significant savings in terms of bandwidth required. Both TCP and RTP header compression are usually implemented per interface but can also be implemented using the MQC policy, class and service maps. The major disadvantage of using header compression is that it adds delay to the connection. With TCP compression this may not be a problem, but with traffic that uses RTP compression, this can become a big issue, should the end-to-end delay budget already be high.

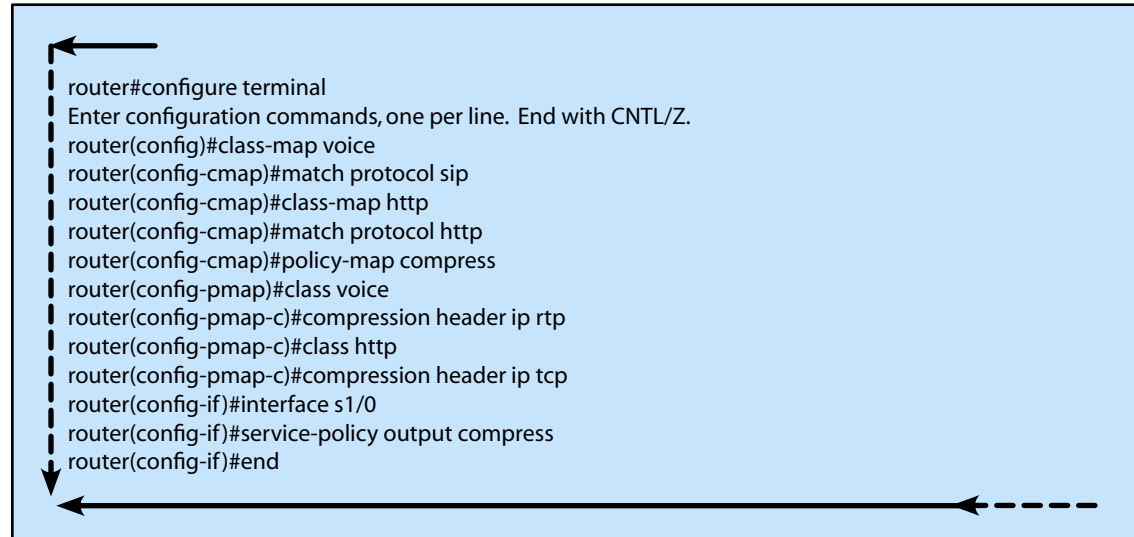
Payload Compression

Payload compression is another option which can be used to save traffic bandwidth. Cisco supports three different payload bandwidth options. Stacker, Predictor, and Microsoft Point-to-Point Compression (MPPC). Stacker compression is quite CPU intensive but does not require a large amount of memory and is supported in some Cisco hardware. Stacker also works with almost any layer 2 point-to-point encapsulation. Predictor compression is not as CPU-intensive but requires higher memory intensity. Predictor only works on PPP and LAPB encapsulations. MPPC is only supported on PPP encapsulations and is used only between a Cisco device and a Microsoft client.

Compression Configuration

Header Compression Configuration

The following example shows how TCP and RTP header compression are configured:



```
router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
router(config)#class-map voice
router(config-cmap)#match protocol sip
router(config-cmap)#class-map http
router(config-cmap)#match protocol http
router(config-cmap)#policy-map compress
router(config-pmap)#class voice
router(config-pmap-c)#compression header ip rtp
router(config-pmap-c)#class http
router(config-pmap-c)#compression header ip tcp
router(config-if)#interface s1/0
router(config-if)#service-policy output compress
router(config-if)#end
```

Figure 51 - RTP/TCP Compression Example

Configuration

ppp multilink

The **ppp multilink** command is used to enable PPP multilink on an interface.

Syntax:

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

ppp multilink fragment delay

The **ppp multilink fragment delay** *milliseconds* command is used to specify the maximum time for transmission of a fragment. The *milliseconds* parameter is used to set the maximum time in milliseconds. By default, this is set to 30ms for interleaved interfaces and interfaces with various bandwidths.

Syntax:

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


ppp multilink interleave

The **ppp multilink interleave** command is used to enable MLP interleaving.

Syntax:

```
router(config-if)#ppp multilink interleave
```

ppp multilink group

The **ppp multilink group** *group-number* command is used to specify the MLP group number. The *group-number* parameter is used to indicate the group number. This number can be any number other than zero.

Syntax:

```
router(config-if)#ppp multilink group group-number
```

frame-relay traffic-shaping

The **frame-relay traffic-shaping** command is used to enable Frame-Relay traffic shaping (FRTS) on an interface.

Syntax:

```
router(config-if)#frame-relay traffic-shaping
```

frame-relay class

The **frame-relay class** *name* command is used to assign a Frame-Relay map-class to an interface. The *name* parameter is used to specify the name.

Syntax:

```
router(config-if)#frame-relay class name
```

frame-relay fragment

The **frame-relay fragment** *fragment-size* command is used to enable Frame-Relay fragmentation. The *fragment-size* parameter is used to specify the size of the fragment.

Syntax:

```
router(config-map-class)#frame-relay fragment fragment-size
```

frame-relay traffic rate

The **frame-relay traffic-rate** *average* [*peak*] command is used to configure FRTS traffic shaping parameters. The *average* parameter is used to specify the average bit rate in bits per second. The *peak* parameter is used to specify the peak bit rate in bits per second.

Syntax:

```
router(config-map-class)#frame-relay traffic-rate average [peak]
```

frame-relay adaptive-shaping

The **frame-relay adaptive-shaping** {**becn** | **foresight** | **interface-congestion** [*queue-depth*]} command is used to configure frame-relay adaptive shaping in map-class mode. The **becn** parameter is used to adapt traffic based on the BECN congestion notifier in the Frame-Relay traffic. The **foresight** parameter is used to adapt traffic based on the information gathered through Cisco Foresight. The **interface-congestion** [*queue-depth*] parameter is used to adapt traffic based on the depth of the interface queue. The default is 0 packets and *queue-depth* can be from 0 to 40.

Syntax:

```
router(config-map-class)#frame-relay adaptive-shaping {becn | foresight |  
interface-congestion [queue-depth]}
```

Domain 9 - QoS Best Practices

- Explain the QoS requirements of the different application types.
- Explain the best practice QoS implementations and configurations within the campus LAN.
- Explain the best practice QoS implementations and configurations on the WAN customer edge (CE) and provider edge (PE) routers.

QoS Application Type Requirements

Voice

Fundamentally voice is the most demanding of application types. This is because of its traffic consistency. Because voice traffic requires a consistent flow of traffic and a specific rate, it is one of the most demanding application types to work with. Once a voice call has been setup, its bandwidth requirements are relatively fixed around a specific bandwidth for the duration of the call. This means that if once a call has been setup, the bandwidth cannot be maintained, the quality of the call suffers, and can possibly affect whether the call continues at all.

Voice also has other requirements which must be maintained which determine the overall quality of the call. These include delay, jitter and packet loss. As discussed earlier in the manual, these two factors can greatly affect the quality of the call. Delay affects the quality of the call by delaying the voice traffic. If voice traffic is delayed too much, it would be hard to maintain a normal conversation as there would always be an extended wait before the voice was received. Jitter affects voice traffic through the varying of delay. If the delay of a packet is constantly changing it makes it hard for the receiver to maintain a consistent call

experience. This is why de-jitter buffers exist. The effects of packet loss are much more obvious. If traffic does not get from one end to the other, the voice conversation cannot continue.

Because of these factors, Cisco has recommended the following reference ranges:

Delay	Jitter	Packet Loss
Less than 150 ms	Less than 30 ms	Less than 1 percent

Video

Video traffic is very similar to voice traffic in terms of requirements: the same delay, jitter and packet loss are expected. However, video is a bit different from voice because its bandwidth can vary during the video. This delay is caused by how the video codecs encode the video. Many of these codecs are setup to initially send the whole frame of video. Then after the frame has been established, only the changes in the video (movement) are sent. Depending on the amount of changes during the course of the video, the bandwidth varies. Because of this change, another category is required: amount of bandwidth fluctuation allowed.

Delay	Jitter	Packet Loss	Bandwidth Fluctuation
Less than 150 ms	Less than 30 ms	Less than 1 percent	< 20 percent

Data

Ultimately, compared to voice and video, data is rather tolerant of all types of network problems. This is because moderate amounts of delay, jitter, packet loss and bandwidth fluctuation do little to affect it. Traditionally, data traffic was considered best effort on a network because it was able to take a secondary position to voice and video traffic. However, there are different types of data traffic and, because of this, there have been a number of categories which have been established. These include mission critical, transactional, best effort and scavenger. Mission critical traffic is the core traffic which goes over the network. These applications tend to be TCP-based and typically are very tolerant of delay and jitter but very bandwidth intensive. Transactional traffic is that traffic which is typically sent between a database client and server. This includes any type of database or any type of application which uses a database server. This type of traffic tends to be more sensitive to delay and jitter but is less affected by bandwidth because it tends to send smaller packets. Best effort traffic on a network is typically the web, email and ftp traffic. This type of traffic is very tolerant to delay, jitter, packet loss and bandwidth restrictions. The scavenger type of traffic tends to be the traffic which is the least important traffic on the network. This tends to include peer-to-peer traffic like bitTorrent.

Campus QoS Recommendations

Cisco recommends the hierarchical model in campus implementations. This includes a distinctive core, distribution and access layer. The core part of the network should be configured to simply respond to QoS information which has already been marked on the traffic. The core should not be marking traffic. A typical core layer is the least likely to be congested. If it is, a type of scheduler is recommended to prioritize traffic sent. The distribution layer is typically used to connect each different part of the network, be it buildings or floors. The access layer connects individual computers and servers into the network.

When dealing with QoS configuration on a campus, a couple of questions should be asked first. At what point in the network can the devices be trusted? If the network is completely controlled by a central team then the whole network is trusted. If not, different parts of the network need to be assigned trust assignments. If the access layer equipment in a network is expected to potentially abuse QoS markings, the trust boundary should be setup at the distribution layer switches. What this means in terms of QoS is that the classification and marking of traffic needs to be done on a trusted piece of equipment and should be done as close to the source as possible. Another question that needs to be asked is, what type of traffic goes across this network? This should be used to devise a list of QoS settings which are used in marking. Some of the Cisco recommendations are listed earlier in this manual.

Customer Edge (CE) to Providers Edge (PE) QoS Recommendations

The transition from a customer piece of equipment to a provider piece of equipment can be a sensitive part of the network. This is because typically neither trusts each other's markings. Because of this, reclassification typically occurs here. However, there are other issues which also tend to occur at this point in the network. These include potential congestion problems. Typically, the amount of bandwidth which is bought from a provider is as little as possible to limit the cost. Because of this, congestion at this point can be a problem. It is here that congestion management and congestion avoidance mechanisms are most beneficial. With the use of these mechanisms, the amount of effect from congestion at this point in the network can be limited.

Another type of configuration which can be done at this point is the development of a Service Level Agreement (SLA). This is an agreement between the customer and the provider which details the different levels of service which must be allowed. If an SLA is setup between a customer and provider, traffic marking can be used to distinguish the different types going into the provider network. These markings may or may not be the same as those used across the internal customer network. If this is the case the markings of traffic going from the customer network into and out of the provider network must be changed.