Exam Manual

LearnSmart

CISCO (642-845) ONT

Smarter Training

This LearnSmart exam manual covers the most important topics on the Optimizing Converged Networks exam (ONT - 642-845). By studying this manual, you will become familiar with an array of exam-related content, including:

- Describing Cisco VoIP implementations
- Describing QoS
- Describing DiffServ QoS implementations
- And more!

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

ONT (642-845)
LearnSmart Exam Manual

Product ID: 11503
Production Date: July 19, 2011

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.

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

The Cisco Certified Network 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 Network Professionals. These four exams, created by Cisco Systems, are extremely difficult and not to be taken lightly. They cover a myriad of topics, in particular - the ONT(Optimizing Converged Cisco Internetworks) exam covers all the way to the most detailed analysis of routing packets wireless and voice over IP networks, including quality of service and other complex network concepts. ONT 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 ONT 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 ONT products, the LearnSmart ONT 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 - Describe Cisco VoIP Implementations

Advantages of VoIP Over Traditional Switches

Voice over IP networks have many advantages over traditional circuit switched voice networks. These advantages include improved bandwidth utilization, consolidated network expenses, and unified messaging integration.

Traditional circuit switched networks utilize a complete 64-kpbs channel; within T carrier networks this is called a DS0. Within a T-1 there are 24 DS0 channels equaling 1.544 Mbps. Packet based voice like VoIP allows more efficient use of this space through compression. This compression allows a number of channels of voice to be transported over a single DS0. The highest level of compression that is currently used is 8:1; meaning a single DS0 can transport 8 channels of voice.

Up until the creation of VoIP, voice and data networks were separate. VoIP enables these two networks to be consolidated into one. Cost is the obvious advantage of having a consolidated network. Having to design and maintain two separate networks is cost prohibitive when it can be done with only one.

VoIP enables unified messaging integration. A phone can become not only a way to talk to someone but also the ability to integrate with video service, database services and several other technologies which enable easier communications throughout a company.

VoIP Components

Like normal telephony systems, VoIP has a number of components which are used to communicate in various ways. These components include phones, gateways, gatekeepers, call agents, Multipoint Control Units (MCU), Digital Signal Processors (DSP) and various application and database servers.

Phones	Just like normal telephony systems, phones are used to communicate via voice. However, with VoIP and Video over IP these phones are extended to not only provide audio communication but also provide video communications.
Gateways	A gateway is a device which is used to communicate between two different types of network. In VoIP applications, this can be used for multiple things including connecting analog phone systems to VoIP systems or connecting from a PBX into a VoIP network. i.e. Using a standard phone over IP (Vonage).
Gatekeepers	Gatekeepers are used for two purposes, to perform some call routing and call lookups (IP to Phone Number), and to perform Call Admission Control (CAC).
Call Agents	Call Agents are used when a central control model is used; this essentially means that call administration is done via a central location. This includes call routing, translations, call setup and teardown among other things.
Multipoint Control Units (MCU)	MCU's are used in conference communications, specifically the MCU is used to combine multiple streams of traffic and join them together seamlessly.

Table continued on next page

Digital Signal Processors (DSP)	Within VoIP, The DSP is used heavily. The DSP's in a VoIP network are used to convert analog signals to digital signals and compress them.
Application and Database Servers	Several different servers can be used in a VoIP implementation; this includes application servers for Call logging and configuration management among others. Database servers can also be integrated with VoIP, this enables XML- based information to be available at the phone level.

Analog & Digital Interfaces

In order to communicate with any network, interfaces must be used. Within a VoIP network there are two different groups of interfaces; analog and digital.

Analog Interfaces

Inside voice networks there are three different types of analog interfaces; Foreign Exchange Office (FXO), Foreign Exchange Station (FXS) and Earth and Magneto (E&M). FXO and FXS interfaces are used with one another, the FXO interface is connected to the telephony switch and the FXS interface connected to the telephone equipment (phone). When a call comes in, the FXO interface is alerted via ring voltage from the switch then the FXO interface tries to transport the signal to the FXS. The FXS is responsible for receiving the signal from the FXO and providing battery, dial tone and other signaling to the telephone equipment. The E&M interfaces are typically used to connect Private Branch Exchanges (PBX) which exists inside offices. The PBX is essentially a small telephony switch that allows different features to be used inside an office environment; these types of features include extensions, forwarding, and conferencing among others. There are five different types of E&M interface; types I through V (1 through 5). The details of each interface are beyond the scope of this manual but types I and V are the most common; Type I is typical in North America and Type V is typical outside North America.

Digital Interface

Within VoIP there are a couple of different digital interfaces; Basic Rate Interface (BRI), T1 and E1 being the main ones. A BRI is used for small office connectivity and provides two channels of voice (64 kbps each), which are called B-Channels and an independent signaling channel (16-kbps), known as the D-Channel. T1 signaling often gets confused because there are two different ways to signal with a T1 interface. The two different ways are Channel-Associated Signaling (CAS) and Common Channel Signaling (CCS). CAS utilizes what is called robbed-bit signaling, this is because T1's are divided into frames which fit into either a Superframe (SF – 12 Frames) or an Extended Superframe (ESF – 24 Frames). Each frame includes 24 timeslots which are used for the 24 T1 channels and each frame includes 8 bits of each channel plus a framing bit. In CAS signaling the 6th and 12th channels have their low order bit "robbed" for use in signaling. CCS signals a completely different way, it utilizes one full 64-kbps channel for signaling and leaves the other 23 channels for traffic; this configuration is typically called a Primary Rate Interface (PRI). Having defined this, BRI is considered a CCS interface utilizing one full 16-kbps signaling channel. E1's all operate the same in a CCS configuration (sometimes they can be incorrectly called CAS). An E1 interface includes a total of 32 channels, with 30 being used for traffic. E1's are split similar to T1's except their framing is a little different. E1's are split into multiframes which include 16 contiguous frames and each frame includes 8-bits of each channel. The 1st and 17th channels are used for frame synchronization and signaling, accordingly. There are no framing bits within an E1 like T1's have.

Traffic Packetization

In order for an analog voice signal to be transmitted across a digital network, it must first be digitized. This is done through a process known as packetization. Packetization has four different phases; sampling, quantization, encoding and compression. As shown in the following figure, packetization takes an analog waveform and converts it into a stream of digital 1's and 0's.

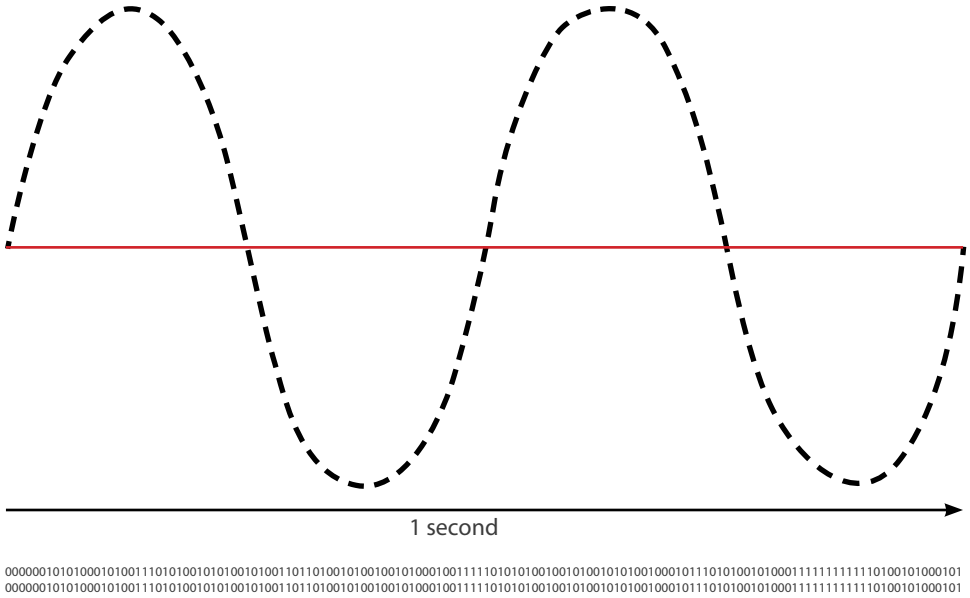


Figure 1 - Packetization

Sampling

The first step in digitizing a signal is to turn the analog wave into something that can be digitized. This is done through sampling which is also called Pulse Amplitude Modulation (PAM); sampling takes slices of the analog wave at consistent intervals. Within sampling, the Nyquist theorem is followed. This theorem states that in order to adequately digitally represent an analog signal, the analog signal must be sampled at a rate of twice the highest analog frequency. Within voice networks the frequency ranges from 300 to 3400 Hz is transmitted, because of simplicity it was decided to sample from 0 to 4000 Hz over digital lines. When following the Nyquist theorem, this means that this signal must be sampled at 8000 Hz which translates to 8000 samples per second.

In order to demonstrate this process the following figures have been created. Instead of showing 8000 Hz sampling the following examples show 40 Hz sampling, to keep it simple.

First an analog signal must be separated into pieces, because 40 Hz sampling is being shown, this analog signal is split into 40 different pieces (samples).

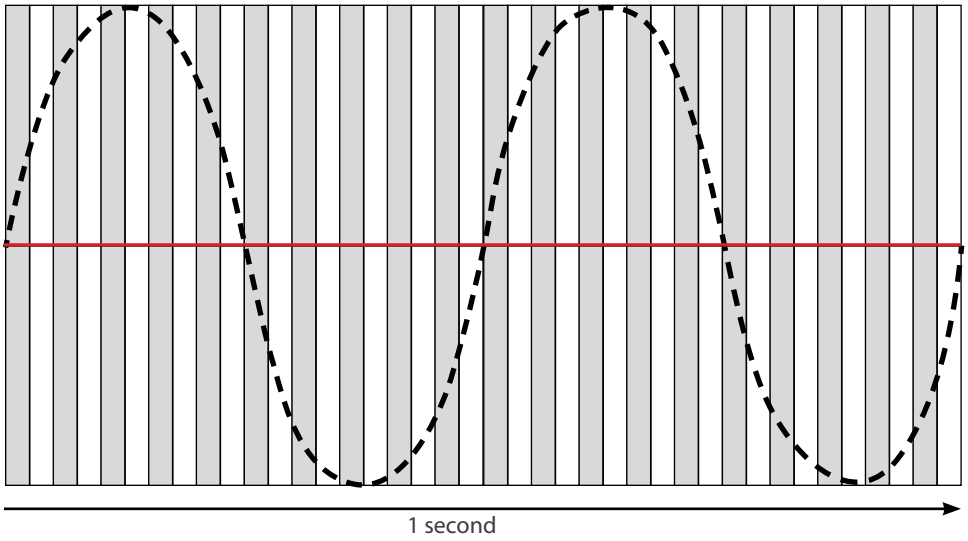


Figure 2 - Separating the Analog Signal

From these, pieces or samples are taken which best represent the analog signal; this is shown in the following figure:

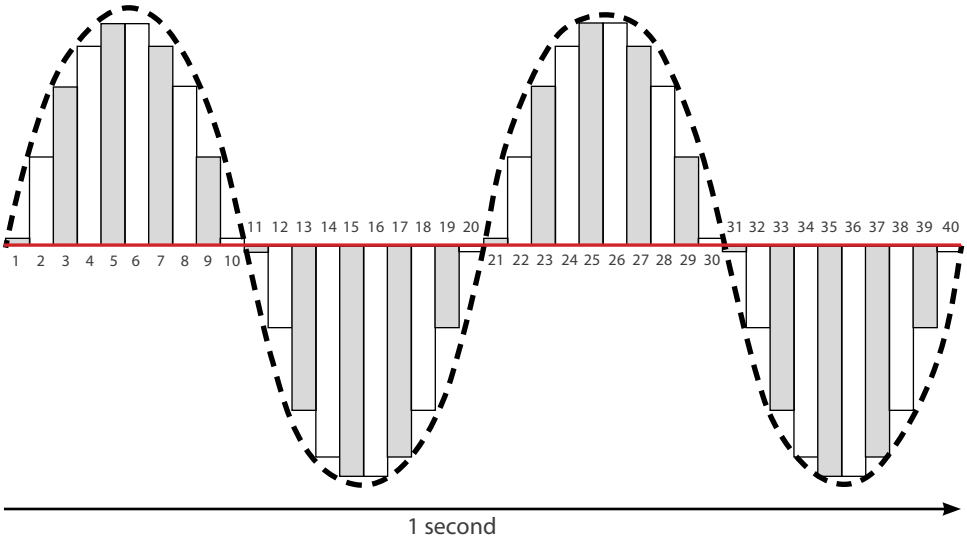


Figure 3 - Creating Samples

From this figure it is seen that the digital samples represent a signal similar to the analog signal being converted. Obviously, the more samples that are taken the more like the original analog signal the digital representation will be.

Quantization

What this stage of processing does is it calculates a mathematical value for each sample taken, this is also called companding. The range of numbers that can be assigned is from -127 to +127, as seen from the following figure each sample is given a number.

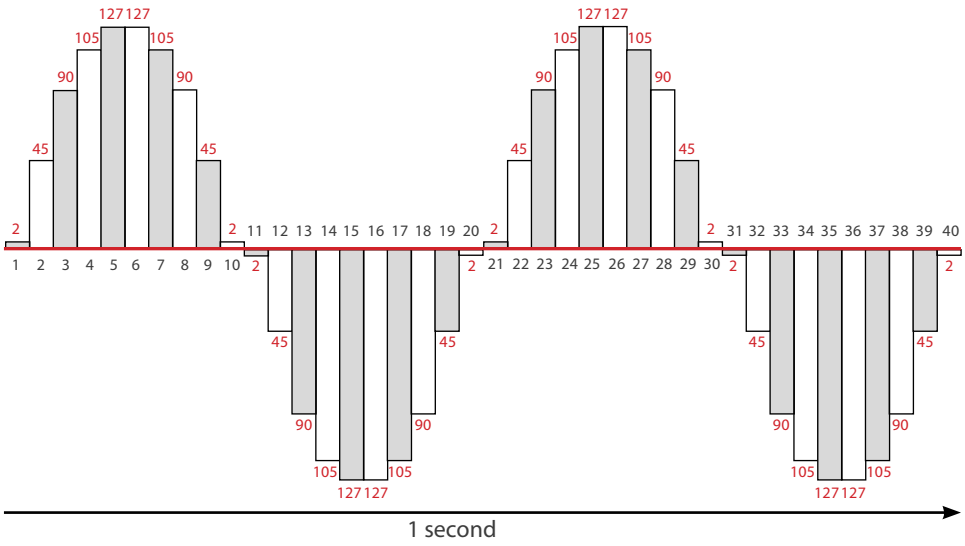


Figure 4 - Quantization

There are two main algorithms which are used to calculate these numbers, μ-law and a-law. μ-law (mu-law) is the formal standard in North America and in Japan; a-law is what is used in the rest of the international community. It is also standard in μ-law countries to convert to a-law in order to communicate with a-law countries.

Binary Encoding and Digital to Digital Encoding

A list of decimal numbers is obtained from quantization. In order to make these numbers into a stream which can be transmitted digitally as binary, encoding is needed. The encoding process takes each number and converts it into a 7-bit binary number with the 1st bit being used to denote the sign (or *polarity*) with 1 meaning negative and 0 meaning positive, the 2nd, 3rd, and 4th bits signifying the *segment*, and the 5th, 6th, 7th and 8th bits signifying the *step*. Once the signal is converted to binary it is run through a digital to digital conversion process which shapes the signal for transmission. The following figure shows this process:

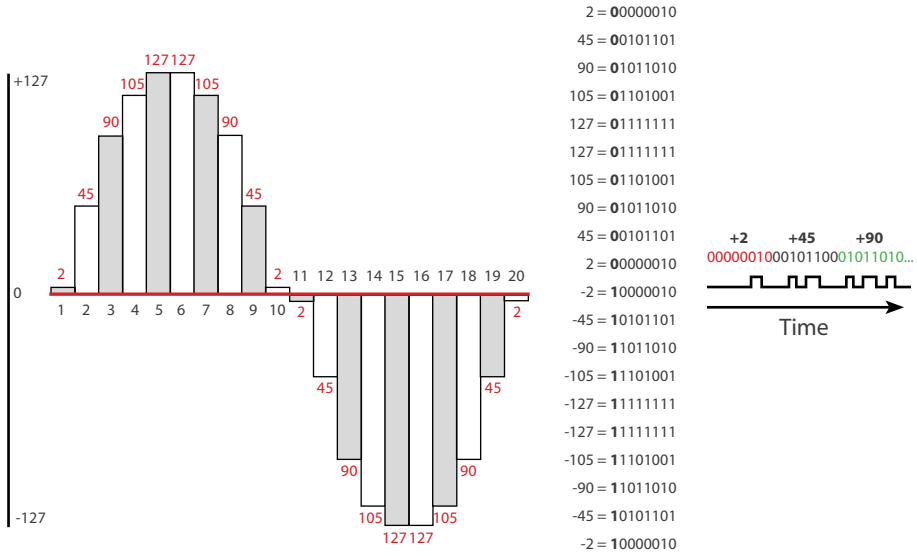


Figure 5 - Binary and Digital-to-Digital Encoding

Compression

Compression is the main thing that has driven the use of VoIP. Through the use of compression, many voice calls can be conducted in the same amount of bandwidth as one had in the past. There are a couple of compression standards (International Telecommunication Union – ITU); these are called codec (Coder – Decoder) standards. The following table lists the standard codecs that are in use today:

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 Exited 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

Figure 6 - ITU Voice Codecs

Digital Signal Processor (DSP)

The DSP within a VoIP network completes the duties required for conversion of the analog signal as well as the duties to perform compression of the signal (analog or digital). Specifically, the DSP performs the voice termination (Analog-to-Digital (A-D) Conversions, Digital-to-Analog (D-A) Conversions, Compression, Transcoding, Conferencing, and echo cancelation among other tasks. The conversion process used in termination is detailed in the previous section. Transcoding is the process of converting from one codec to another. Conferencing can also be done by the DSP. Echo cancelation is done to reduce the amount of echo heard by the call participants, this is done to make the conversation as "normal" as possible.

Call Stages

Call Setup

As the name implies this is the stage that sets up the call. This stage can be broken down into the initial call request, call proceeding, call progress, alerting and the connection. The initial call request involves looking up the telephone to IP conversion for call routing, and trying to send a request to the called party. Now a call can fail at this point for a number of reasons, including no response to the lookup (don't know where to go for the number), no resources available (no bandwidth or trunk space for the call), or no response from any of the nodes inline to the called party. Call proceeding notifies the calling party that the call is going through. Call progress notifies the calling party of the status of the call. Call alerting tells the calling party the status through the use of sounds (busy, fast busy…) and the connection (or Connect) is when the call has been answered, parameters are negotiated and the connection is established. The main negotiated parameters include ports used for the call and the codec.

Call Teardown

Call teardown is easy, a request is sent from either side of the connection to terminate. The switch notifies all the switches inline from the calling to the called party to teardown the connection. Once this is complete there is no connection left.

Call Maintenance

Call maintenance can be done by any part of the voice network. The switch can collect data on all calls and connections to and through it. The phones and other endpoints can collect information about the call including call statistics, quality, delay and jitter among other things.

Call Control Models

Distributed

The distributed call control model is used with H.323 and SIP. Distributed call control works by distributing the duties of the network over several different network components. What this means is that call setup, maintenance, routing, CAC and teardown can be controlled through several different devices.

Central

The central call control model is used with MGCP. Central call control works by having the call setup, maintenance, routing, CAC and teardown duties controlled by a Call Agent (CA). The CA is typically at a central location and controls a number of endpoints. The endpoints retain the digitization, encapsulation and transportation duties. Once a call has been setup between the 1st endpoint and the CA and the 2nd endpoint and the CA, the call continues between the 1st and 2nd endpoint independent of the CA until call teardown.

Call Signaling

H.323

H.323 specifies a group of protocols which are used to connect devices together through a distributed model, specifically for multimedia, to include voice and video. With H.323 there are four phases to creating a connection, this includes an admission request, connection setup, capabilities exchange and the opening of the media.

In the admission request phase the endpoint or gateway communicates with the gatekeeper via H.225 over UDP. The gatekeeper checks to see if it knows how to route the call requested, if the call route exists the gatekeeper checks if a path is available from the 1st to the 2nd endpoints with available resources. If the resources are available then the gatekeeper performs an address translation from phone number to IP address and returns the information to the requesting endpoint.

During the connection setup, the endpoints communicate directly via H.225 over TCP. During this the endpoints take the lookup information and establish a connection to the endpoint. During this connection a setup message is sent from the originating endpoint, if the call is accepted a connect message is exchanged.

During the capabilities exchange the endpoints communicate directly via H.245 over TCP. Capabilities which are exchanged include voice or video communications, codec exchange, compression exchange and coding exchange including others. It is at the end of this phase that the end user is finally notified of an incoming call.

Now the final phase includes the opening of the media via H.225 over TCP. If the end user has answered the call connect messages are used to open the connection.

SIP

Session Initiation Protocol (SIP) is similar to H.323 in that it works as a distributed model and uses several different separate protocols. SIP as opposed to H.323 was developed with the Internet in mind, and as such is text based and looks similar to Hypertext Markup Language (HTML). SIP also is addressed similarly to web pages with a URL, this URL looks like sip:far-end-user@testing.com. SIP as a protocol is used to initiate and find the target recipient. Once the recipient is found the Session Announcement Protocol (SAP) takes over by identifying what type of session is trying to be established (voice or video), this is carried over the network with the Session Description Protocol (SDP). SAP and SDP are used to create or change the parameters of a call.

MGCP

The Media Gateway Control Protocol (MGCP) works a little different than the other two protocols. MGCP works by controlling multiple gateways. Within MGCP there are two main components, the Media Gateway (MG) and the Media Gateway Controller (MGC). The MG's are responsible for connecting and translating connections into the network. These can exist anywhere from the end user's house to a central telephone office. The MGC is the call agent and is responsible for call control, for more information on Call Agents see the Central call control model section.

RTP

All voice over IP traffic (not signaling) is carried via UDP; however UDP by itself has some problems that need to be addressed for Voice over IP service to work. UDP is a connectionless protocol and does not have packet sequencing and reordering capabilities or the ability to time stamp. Because of these shortfalls, another protocol, Real-Time Transport Protocol (RTP), was created to run over UDP and provide these capabilities. These capabilities on top of UDP's ability to multiplex traffic make voice over IP technologies work well and keep a high quality level. RTP is used with all of the above signaling protocols. The problem with RTP is that its header adds extra overhead to the voice packet, 12 bytes in total for voice with an additional 60 bytes possible if optional headers are used for other types of media streams. RTP also has a monitoring protocol that works with it called Real-Time Transport Control Protocol – (RTCP). RTCP is used to monitor the RTP session and update the participants of the status of the stream.

RTP header compression (cRTP)

While RTP provides extra needed services that UDP does not, it also provides added overhead. So with RTP the total overhead of an IP packet is 40 bytes, 20 bytes for the IP header, 8 bytes for the UDP header and 12 bytes for the RTP header. What cRTP provides is a way to minimize the header overhead. cRTP does this by assigning a hash to the IP, UDP and RTP headers, which then replaces the IP, UDP and RTP headers completely. Since these headers for the duration of a call are exactly the same, the hash is used to replace them until the headers change, if the headers do change cRTP sends the headers again. This hash without checksum is only 2 bytes saving 38 bytes of overhead per packet, with checksum the hash is 4 bytes saving 36 bytes of overhead per packet. cRTP is applied per link and is required to be run on both sides of the link for it to work correctly. It is the recommendation of Cisco that this only be used on connections below 2 Mbps when cRTP is performed in hardware and below 768-kbps if in software. cRTP does add delay when computing the hashes and should be considered when calculating the overall expected delay. By default, Cisco places a limit of 16 concurrent cRTP sessions per link; this can be changed if the resources are available.

Delay & Jitter

Delay is an important part of voice communications, the more delay there is on a call the less comfortable it will be to continue a normal conversation. Because of this, the ITU created the G.114 standard. This standard states that one-way delay should be limited to 150-ms for the best quality and up to 400-ms for acceptable quality. Anything above 400-ms of one-way delay is considered to be unacceptable and very noticeable. When designing a voice network it is important to keep this overall delay budget in mind, every step along the voice path adds additional delay. As long as this delay can be limited to within these limits the calls will have acceptable quality.

Jitter is a little different; it is the varying amount of delay between voice packets. If nothing is done to mitigate jitter over a line the voice devices receive incoming voice packets at random delayed intervals, this can cause the sound to become choppy. Jitter buffers exist to smooth out these delays by buffering small amounts of traffic.

Call Admission Control (CAC)

CAC is a vital part of any voice network; it limits the amount of resources which are currently being used on the network. CAC does this by keeping track of how many resources are available on the network. When a call is requested it checks with CAC to see if there are resources available to carry the call. If these resources are available the call is allowed to go through, if these resources are not available then the call is sent back with a message of unavailable resources. On the public phone system everyone has heard this when they get a fast busy or an "all circuits are busy" message.

Bandwidth Calculation

Bandwidth calculation is one of the most overlooked and least well explained topics about VoIP. When planning a VoIP network, calculations must be made to find the true bandwidth required for the number of VoIP calls needed concurrently. Because this calculation can be complex without knowing all the numbers the following information is assumed in most calculations:

Codec	Codec Bit Rate	Voice Payloads (Bold Default) in Bytes	Codec Sample Size in Bytes	Codec Sample Interval
G.711	64-kbps	80, **160**	80	10ms
G.722	64-kbps	80, **160**, 240	80	10ms
G.723.1	5.3-kbps	**20** to 220 in multiples of 24	20(19.875)	30ms
G.723.1	6.3-kbps	**24** to 216 in multiples of 24	24(23.625)	30ms
G.726	16-kbps	20 to 220 in multiples of 20 (**40**)	10	5ms
G.726	24-kbps	30 to 210 in multiples of 30 (**60**)	15	5ms
G.726	32-kbps	40 to 200 in multiples of 40 (**80**)	20	5ms
G.728	16-kbps	10 to 230 in multiples of 10 (**40**)	10	5ms
G.729	8-kbps	10 to 230 in multiples of 10 (**20**)	10	10ms
G.729A	8-kbps	10 to 230 in multiples of 10 (**20**)	10	10ms

Keep in mind that the following calculation is used to make up these numbers:

Memorize these calculations:

$$\text{codec bit rate} = \frac{\text{codec sample size}}{\text{codec sample interval}}$$

codec sample size = codec bit rate x codec sample interval

$$\text{codec sample interval} = \frac{\text{codec sample size}}{\text{codec bit rate}}$$

PPS = (1 second / (sample interval x (Payload / Sample size)))

The amount of overhead the interface introduces also needs to be considered in the bandwidth calculation. For the following examples the following overhead will be used:

- 40 Bytes – 20 Bytes for IP, 8 Bytes for UDP, and 12 Bytes for RTP
- cRTP links will have an overhead of either 2 or 4 Bytes depending on whether a checksum is used
- 18 Bytes for Ethernet connections, includes 4 Bytes for FCS
- 6 Bytes for PPP Connections, includes 2 Bytes for FCS

The following table shows some of these calculations:

Codec	**Codec Bit Rate**	**Voice Payload Size**	**Packets Per Second (PPS)** PPS = (1 second / (sample interval x (Payload / Sample size)))
G.711	64-kbps	160	50
G.722	64-kbps	160	50
G.723.1	5.3-kbps	20	34 (33.$\overline{33}$)
G.723.1	6.3-kbps	24	34 (33.$\overline{33}$)
G.726	16-kbps	40	50
G.726	24-kbps	60	50
G.726	32-kbps	80	50
G.728	16-kbps	40	50
G.729	8-kbps	20	50
G.729A	8-kbps	20	50

The following figures detail bandwidth calculations for several of the main codecs used today:

*G.*711 Sample
160 Bytes x 50 PPS = 8000 Bytes x 8 = 64,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
64,000 bits per second + 23,200 bits per second = 87,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
64,000 bits per second + 18,400 = 82,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
64,000 bits per second + 3,200 = 67,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
64,000 bits per second + 4,000 = 68,000 bits per second

Figure 7 - G.711 Bandwidth Calculation

*G.*722 Sample
160 Bytes x 50 PPS = 8000 Bytes x 8 = 64,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
64,000 bits per second + 23,200 bits per second = 87,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
64,000 bits per second + 18,400 = 82,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
64,000 bits per second + 3,200 = 67,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
64,000 bits per second + 4,000 = 68,000 bits per second

Figure 8 - G.722 Bandwidth Calculation

*G.*723.1 (5.3-kbps) Sample
20 Bytes x $33.3\overline{3}$ PPS = $666.6\overline{6}$ Bytes x 8 = 5,333.33 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x $33.3\overline{3}$ = 1933.33 Bytes x 8 = 15,466.66 bits per second Ethernet overhead
5,333.33 bits per second + 15.466.66 bits per second = 20,800 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x $33.3\overline{3}$ = 1,533.33 Bytes x 8 = 12,266.67 bits per second
5,333.33 bits per second + 12,266.67 = 17,600 bits per second

10 (4 cRTP-Checksum + 6 PPP) x $33.3\overline{3}$ = 333.33 Bytes x 8 = 2,666.67 bits per second
5,333.33 bits per second + 2,666.67 = 8,000 bits per second

Figure 9 - G.723.1 (5.3-kbps) Bandwidth Calculation

*G.*726 (16-kbps) Sample
40 Bytes x 50 PPS = 2000 Bytes x 8 = 16,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
16,000 bits per second + 23,200 bits per second = 39,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
16,000 bits per second + 18,400 = 34,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
16,000 bits per second + 3,200 = 19,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
16,000 bits per second + 4,000 = 20,000 bits per second

Figure 10 - G.726 (16-kbps) Bandwidth Calculation

*G.*726 (24-kbps) Sample
60 Bytes x 50 PPS = 3000 Bytes x 8 = 24,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
24,000 bits per second + 23,200 bits per second = 47,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
24,000 bits per second + 18,400 = 42,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
24,000 bits per second + 3,200 = 27,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
24,000 bits per second + 4,000 = 28,000 bits per second

Figure 11 - G.726 (24-kbps) Bandwidth Calculation

*G.*726 (32-kbps) Sample
80 Bytes x 50 PPS = 4000 Bytes x 8 = 32,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
32,000 bits per second + 23,200 bits per second = 55,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
32,000 bits per second + 18,400 = 50,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
32,000 bits per second + 3,200 = 35,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
32,000 bits per second + 4,000 = 36,000 bits per second

Figure 12 - G.726 (32-kbps) Bandwidth Calculation

*G.*728 Sample
40 Bytes x 50 PPS = 2000 Bytes x 8 = 16,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
16,000 bits per second + 23,200 bits per second = 39,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
16,000 bits per second + 18,400 = 34,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
16,000 bits per second + 3,200 = 19,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
16,000 bits per second + 4,000 = 20,000 bits per second

Figure 13 - G.728 Bandwidth Calculation

*G.*729 Sample
20 Bytes x 50 PPS = 1000 Bytes x 8 = 8,000 bits per second

58 (40 IP/UDP/RTP + 18 Ethernet) x 50 = 2,900 Bytes x 8 = 23,200 bits per second Ethernet overhead
8,000 bits per second + 23,200 bits per second = 31,200 bits per second total

46 (40 IP/UDP/RTP + 6 PPP) x 50 = 2,300 Bytes x 8 = 18,400 bits per second
8,000 bits per second + 18,400 = 26,400 bits per second

8 (2 cRTP-No-Checksum + 6 PPP) x 50 = 400 Bytes x 8 = 3,200 bits per second
8,000 bits per second + 3,200 = 11,200 bits per second

10 (4 cRTP-Checksum + 6 PPP) x 50 = 500 Bytes x 8 = 4,000 bits per second
8,000 bits per second + 4,000 = 12,000 bits per second

Figure 14 - G.729 Bandwidth Calculation

Now these samples calculate the bandwidth requirements of only the RTP traffic, which is essentially for voice throughout the call. On top of this it can be estimated that another 5% be added for signaling traffic overhead.

Effects of Tunneling

Another thing that was not part of these calculations was whether the VoIP traffic was going to be tunneled from one side of the call to the other. Tunneling, as the name implies, tunnels a packet through a network, usually for security purposes. When a packet is tunneled, another header is added separately from the main packet header to be used by the tunneling protocol for routing. As with codecs there are a various amount of tunneling protocols, the following is a short list of the main tunneling protocols used and their overheads:

Protocol	Overhead (Header)
IPSec Transport (DES/3DES)	30 to 37 Bytes
IPSec Transport (AES)	38 to 53 Bytes
IPSec Tunnel (DES/3DES)	50 to 57 Bytes
IPSec Tunnel (AES)	58 to 73 Bytes
L2TP	24 Bytes
GRE	24 Bytes
MPLS	4 Bytes
PPPoE	8 Bytes

Figure 15 - Tunnel Overhead

Voice Activity Detection (VAD)

VAD is a way to reduce the amount of bandwidth used in a call by detecting the silence in a call. VAD works by waiting for silence on a call typically one way (on hold), when this happens the device stops sending traffic in that direction until the end of the silence. It is estimated that using VAD provides up to 35% bandwidth savings.

Mean Opinion Score (MOS) & Perceptual Speech Quality Measure (PSQM)

MOS & PSQM are a way to measure the quality of a voice call. The following chart shows MOS scores of a couple of the main codecs. These scores can vary a little depending on the codec implementation.

Codec	MOS Score (1-Bad to 5-Excellent)
G.711	4.2
G.723.1 (5.3)	3.7
G.723.1 (6.3)	3.9
G.726 (32)	3.85
G.728 (16)	3.61
G.729	3.92

Callmanager Functions

Cisco Call Manager (CCM) is the main call processing software. CCM works with all of the call signaling protocols discussed in this manual and acts as the Call Agent with MGCP. The following are the main features of Cisco Callmanager:

Call Processing	Call routing, signaling, accounting, bandwidth management and Class of Service capabilities.
Dial Plan Administration	CCM handles the dial plan administration centrally.
Call Signaling	When acting as a CA for MGCP CCM controls the signaling for the gateways.
Phone Feature Administration	IP Phone configuration is controlled through CCM centrally.
Directory and XML Services	These services can be centrally administrated by CCM.
Programming API	CCM provides an API to be used by external applications.

Enterprise Deployment Models

In order to make enterprise deployment easier, Cisco has laid out a couple of deployment models that can be followed when implementing VoIP and Cisco Call Manager. These include the following:

Deployment Model	Description
Single-Site Model	All phones are limited to one location. A CCM cluster can exist at this one location which controls all phone calls and all external access to the Public switched telephone network (PSTN).
Multisite with Centralized Call Processing Model	All phones over many sites are centrally administrated at one site. In this type of deployment a CCM cluster exists at the one central site. If the connection goes down to the central site Cisco has a feature called Survivable Remote Site Telephony (SRST), which allows the local gateway to take over these duties until the connections are reestablished.
Multisite with Distributed Call Processing Model	All phones over all sites are managed at each site; a CCM cluster exists at each site to control traffic and to have control resources.
Clustering over WAN	This type of configuration is similar to the Centralized model, but instead of having the whole cluster at one main site several components of the cluster are spread over many sites. A total of one cluster exists.

Domain 2 - Describe QoS Considerations

QoS Basics

Quality of Server (QoS) is a group of technologies which provide a way to specify different quality parameters to different types of traffic. Essentially, QoS is needed in today's networks because more and more they are becoming data and voice converged networks, meaning that both data and voice flow over the same network. This of course makes both data and voice traffic influential over each other. Some traffic like file transfers or web pages are 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 packet arrives late it becomes useless in the audio stream. The follow sections will talk about the four main factors which must be addressed with QoS technologies.

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 then is available on the network then 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 concerned with. 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:

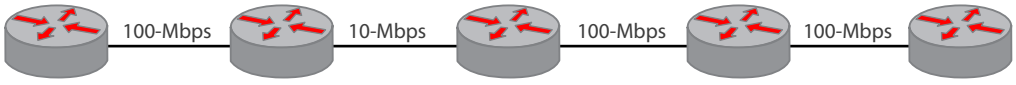


Figure 16 - Bandwidth Example

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

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 five main delay factors are processing delay, queuing delay, serialization delay, and propagation delay.

Processing delay is the amount of time that 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 that 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.

Serialization delay is the amount of time that it takes for a packet to be broken down into layer 2 frames then into layer 1 electric or optical signals.

Propagation delay is the amount of time it takes for a packet to cross the physical medium.

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 out of order packets and reorganizing them back into order. This can only be done correctly however when the overall amount of jitter is minimal. This jitter buffer also adds additional end-to-end delay.

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 it dropped.

Input drop is when a router tried to receive a packet but it's input queue is completely full.

Overruns are what 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 was no level 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 Models

A QoS model is a group of end-to-end capabilities that are 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 which will 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 then a confirmation is sent back to the end device. Once this confirmation is received, then 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; 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. On Cisco IOS this can be done through Committed Access Rate (CAR). WFQ and WRED can also be used with differentiated services.

QoS Implementation

When implementing QoS, there are three main steps that 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 which are using the network. With this information, the network designers will create traffic parameters which are needed for each class of traffic. The traffic classification step is where 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; this 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. This 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 automatically 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 that 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:

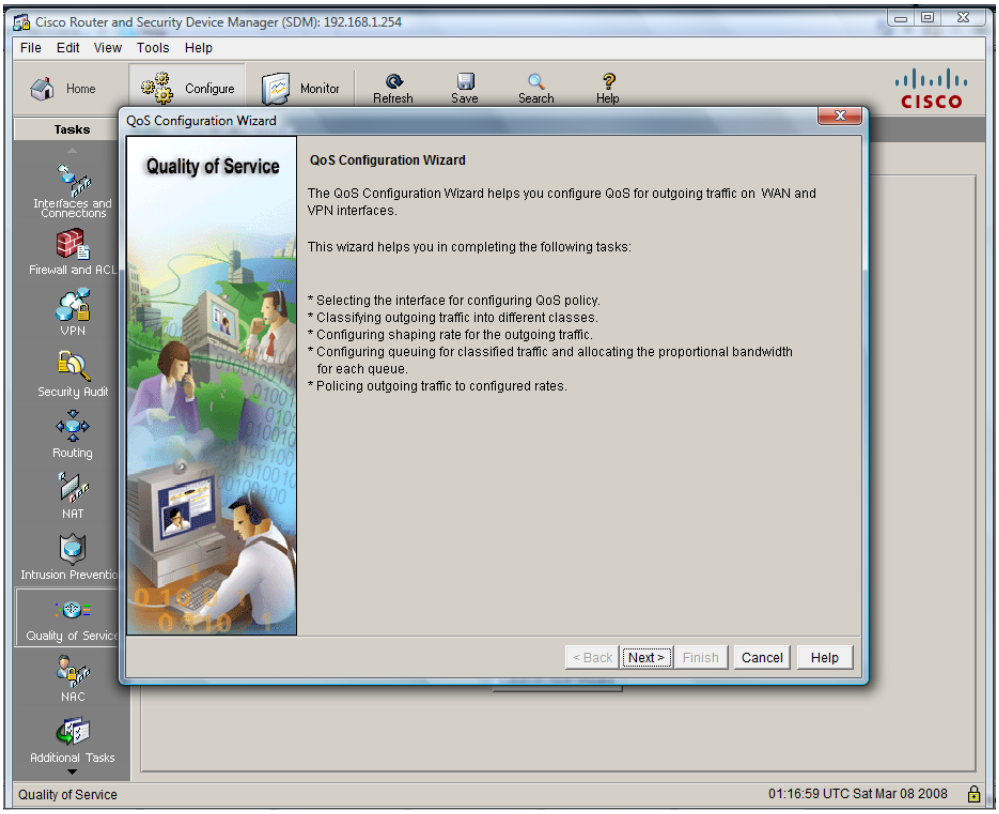


Figure 17 - Cisco SDM QoS Configuration Wizard

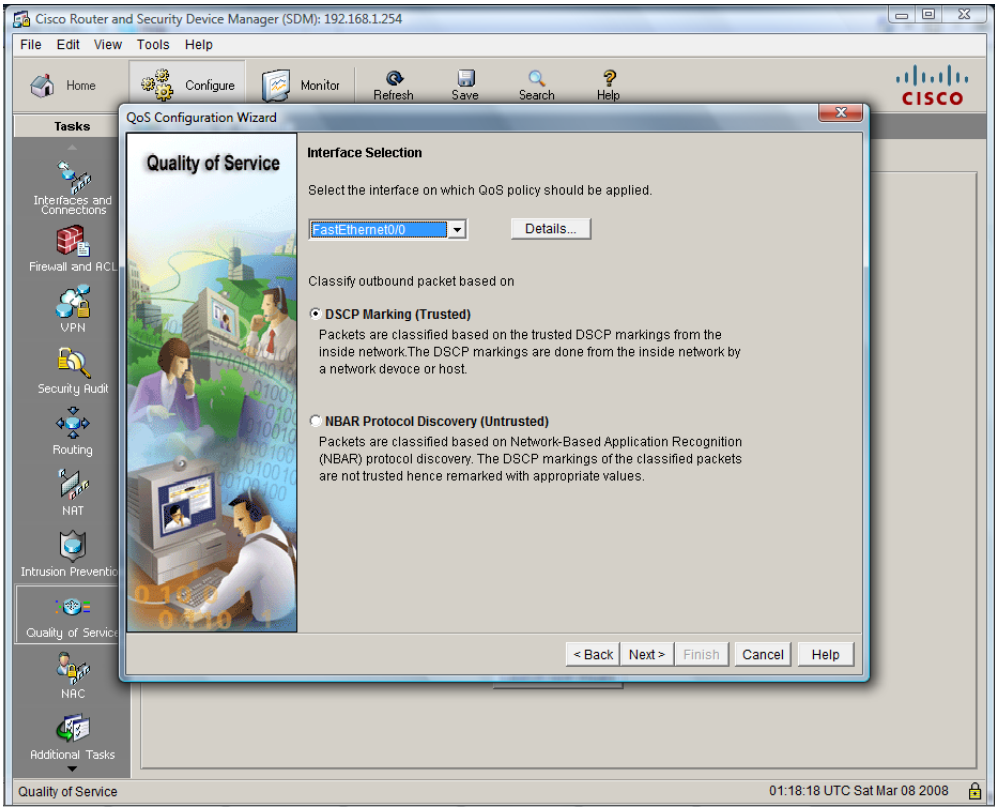


Figure 18 - SDM QoS Interface Configuration

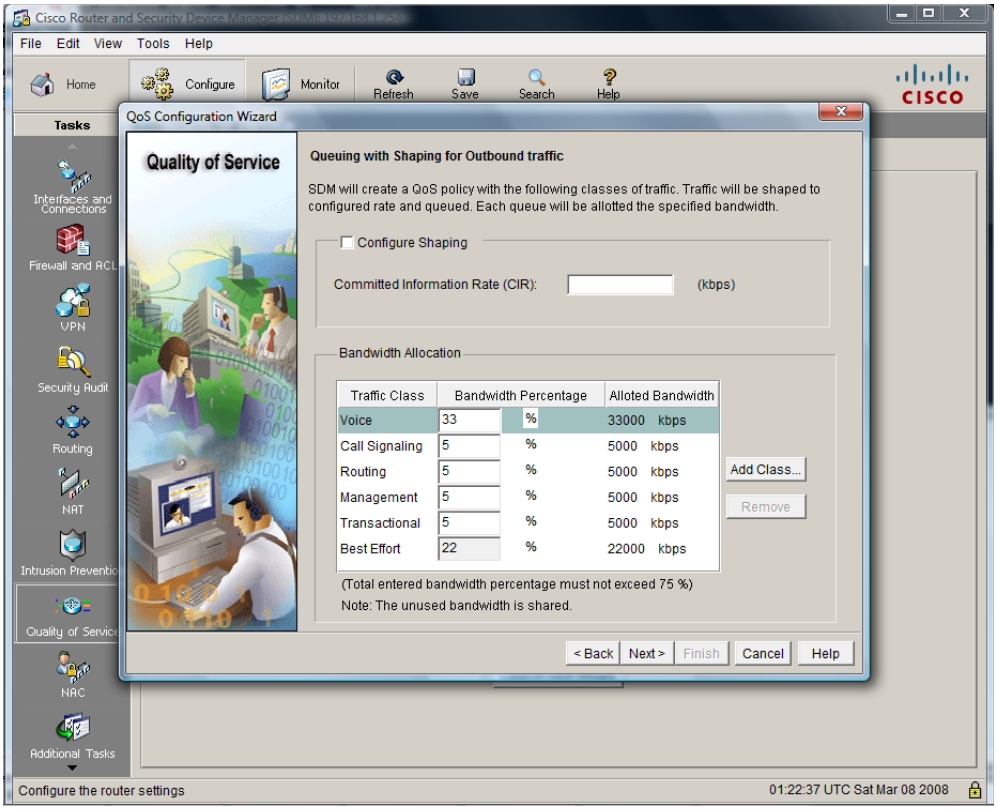


Figure 19 - SDM QoS Queuing and Shaping

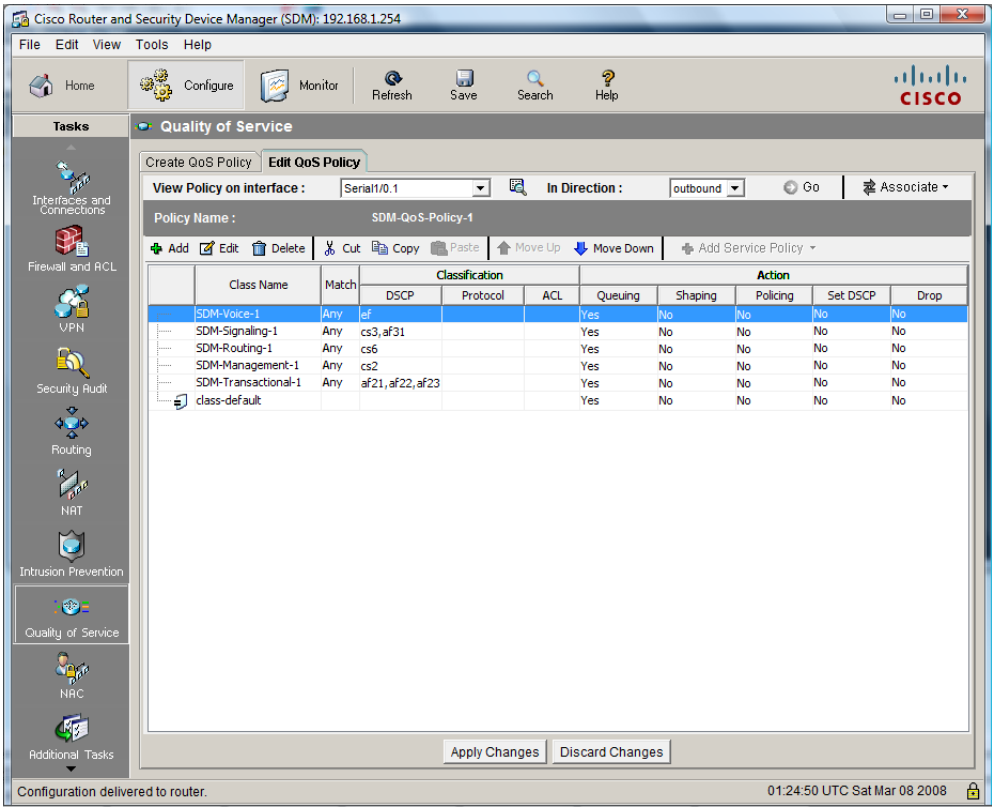


Figure 20 - SDM QoS Edit Policy

Configuration

Here are several of the commands that are used to create and implement QoS policies:

class-map

The **class-map** [**match-all** | **match-any**] *class-map-name* command is used to create a class-map. The **match-all** and **match**-any parameter is used to specify how the following match commands are interpreted. the **match-all** parameter is the default and interprets the match commands with a logical AND. The **match-any** parameter interprets the match commands with a logical OR. The *class-map-name* specifies the name of this class-map.

Sytnax:

router(config)#**class-map** [**match-all** | **match-any**] *class-map-name*

policy-map

The **policy-map** *policy-map-name* command is used to create a policy-map. The *policy-map-name* parameter specifies the name of this policy-map.

Sytnax:

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

service-policy

The **service-policy** {**input** | **output**} *policy-map-name* command is used to apply a policy to an interface. The **input** and **output** parameters specify the direction that the policy will be applied. The *policy-map-name* parameter specifies the name of the policy-map to be applied.

Sytnax:

router(config-if)#**service-policy** {**input** | **output**} *policy-map-name*

match dscp

The **match** [**ip**] **dscp** *dscp-value* [*dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value dscp-value*] command is used to match differentiated service code point (DSCP), Assured Forwarding (AF), and Certificate Server (CS) points. The **ip** parameter specifies to only match IPv4 traffic. The *dscp-value* parameter specifies a dscp value.

Sytnax:

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

match protocol

The **match protocol** *protocol-name* command is used to match a specific traffic protocol. The *protocol-name* parameter specifies the name of the traffic protocol to be matched.

Sytnax:

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

match not

The **match not** *match-criterion* command is used to NOT match a specific match criteria. The *march-criterion* parameter specifies the criterion to be matched, the values to be used here are the same as would be used for the **match** command (i.e. **match not protocol**).

Sytnax:

router(config-cmap)#**match not** *match-criterion*

class

The **class** *name* command is used to assign a class-map to a policy map. The *name* parameter specifies the name of the class-map to be assigned.

Sytnax:

router(config-pmap)#**class** *name*

bandwidth

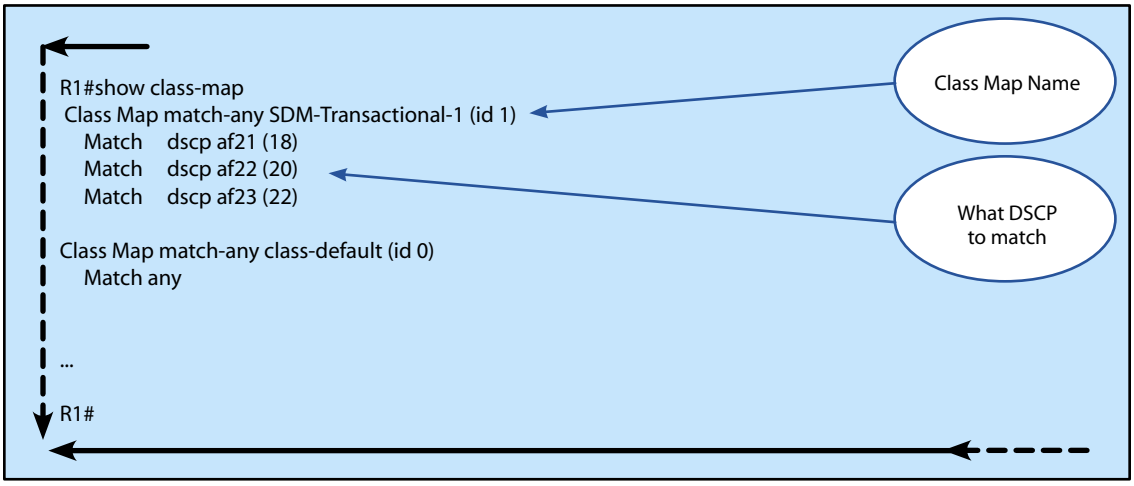
The **bandwidth** *kbps* command is used to specify the bandwidth of an interface; the *kbps* parameter specifies the bandwidth in kilobits per second.

router(config-if)#**bandwidth** *kbps*

Troubleshooting

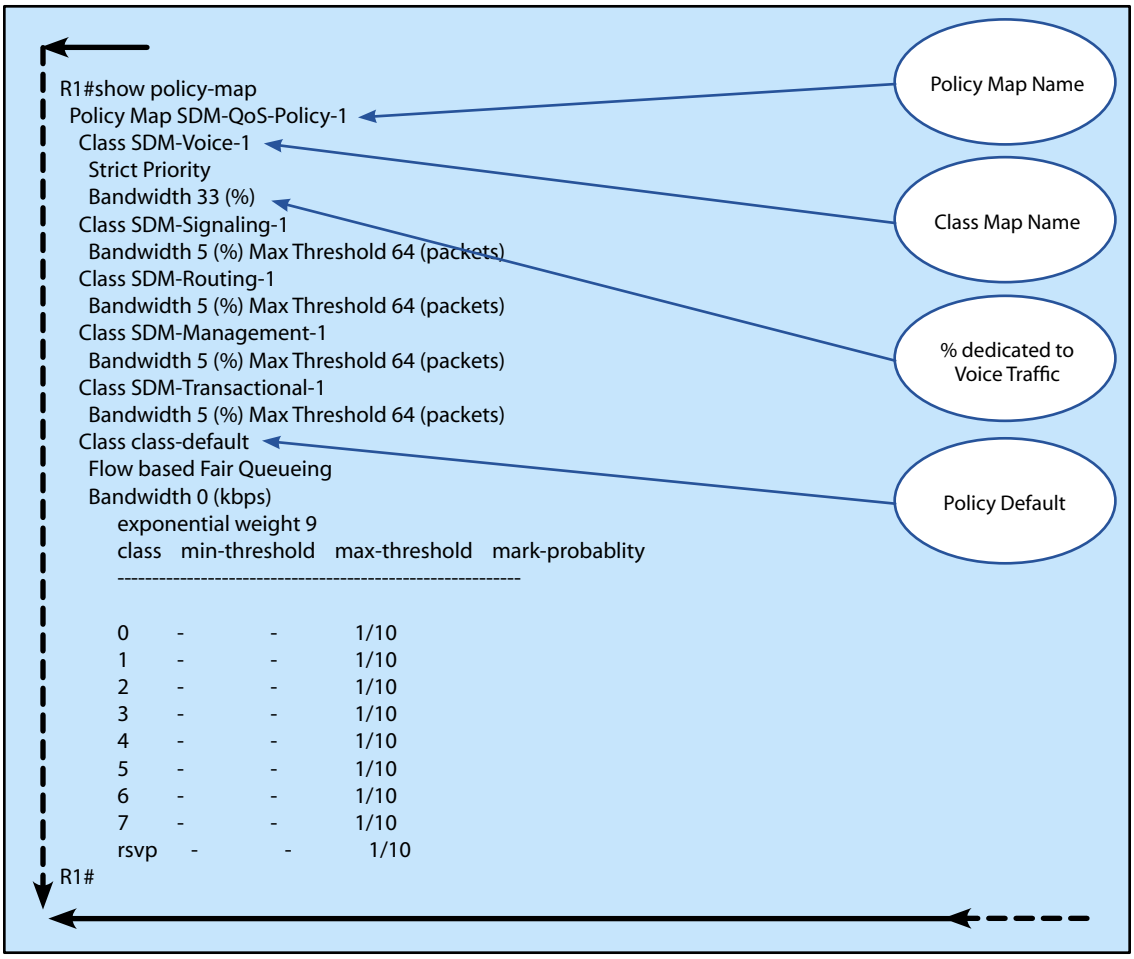
show class-map

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



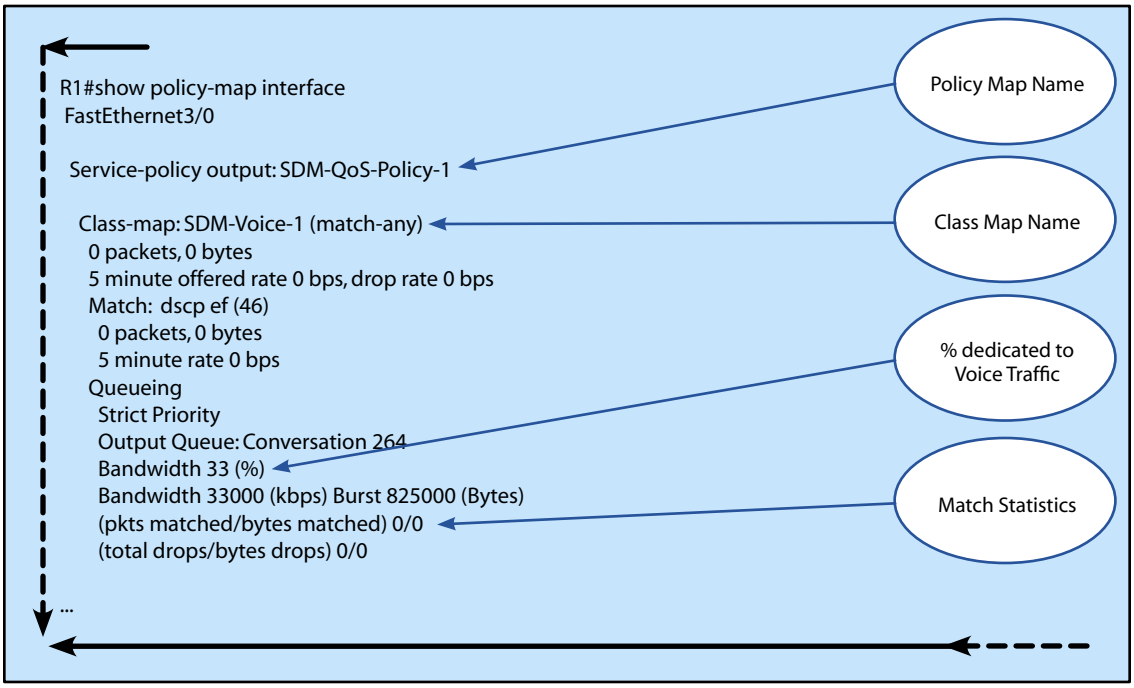
show policy-map

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



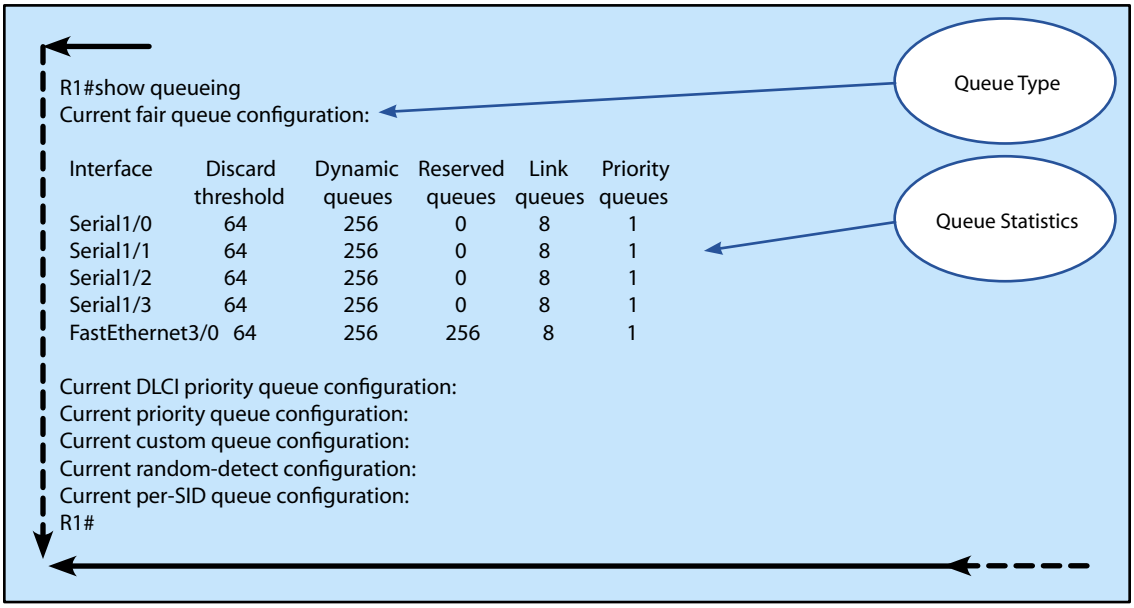
show policy-map interface

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



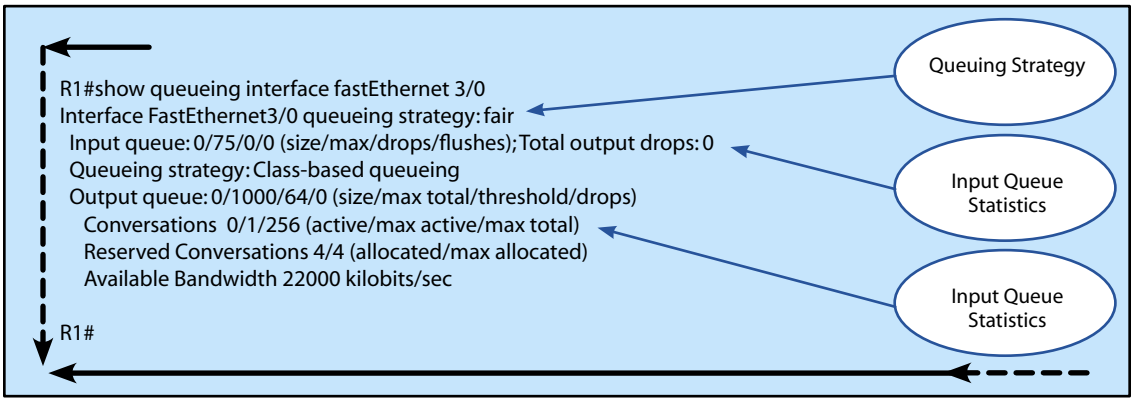
show queuing

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



show queuing interface

This command is used to display information about configured queuing on an interface. The following highlights the most important parts.



Domain 3 - Describe DiffServ QoS Implementations

QoS Primer

What needs to be remembered about Quality of Service is the purpose of it. 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 mechanisms which are used to remedy these problems. There are different types of ways to mark traffic; within networks there are two main ways to do this. 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 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 has the option to use preempt congestion by dropping packets based on a congestion threshold.

This domain will go over the QoS options that are used in DiffServ. The routers themselves can be configured to make QoS decisions based on the information in the traffic but there is no end-to-end QoS mechanism. Cisco routers use a number of different mechanisms to provide QoS. These mechanisms include congestion management solutions and also queuing mechanisms. Congestion avoidance mechanisms include Tail Drop and Weighted Random Early Detection (WRED). Traffic control mechanisms include traffic policing and shaping. Traffic policing is a way to make sure that certain traffic is only allowed to use a specific configured amount of bandwidth; if there is more than the configured traffic then the traffic is dropped. Traffic shaping allows the configuration of the traffic flow and how traffic is allowed to flow across the router. Traffic efficiency mechanisms include header and payload compression and fragmentation (MLP).

Traffic Classification and Marking

The first step in using QoS is a mechanism that is used to separate the traffic into different levels of priority and to mark this traffic so that networking equipment along the path of the traffic will have something to look for. Within QoS this is accomplished through *classification* and *marking*. Classification is the act of looking into the traffic and assigning the traffic into classes based on a number of different parameters.

This classification can be done on almost anything in the traffic. These include 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.

Marking is the action of changing a part of the traffic to mark their classification so that networking equipment along the path can routed based on this classification. This can be done in a number of ways because there are a number of different options for marking. 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 and 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).

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 Figure 21). In order to provide for priorization 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 22). These 3-bits provide for eight different levels of class which can be defined for each Ethernet frame.

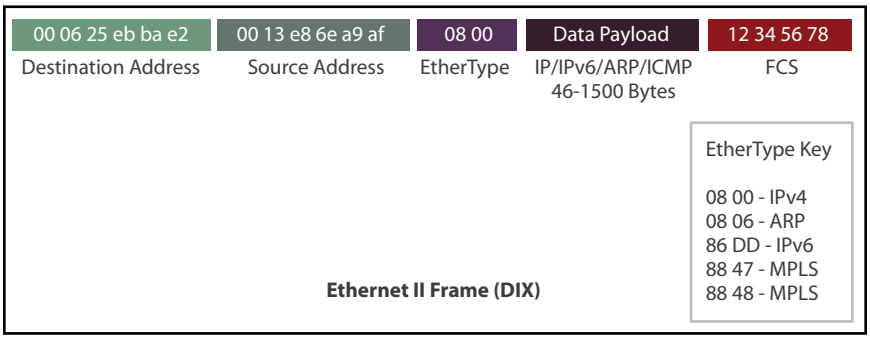


Figure 21 - Ethernet II Frame

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.

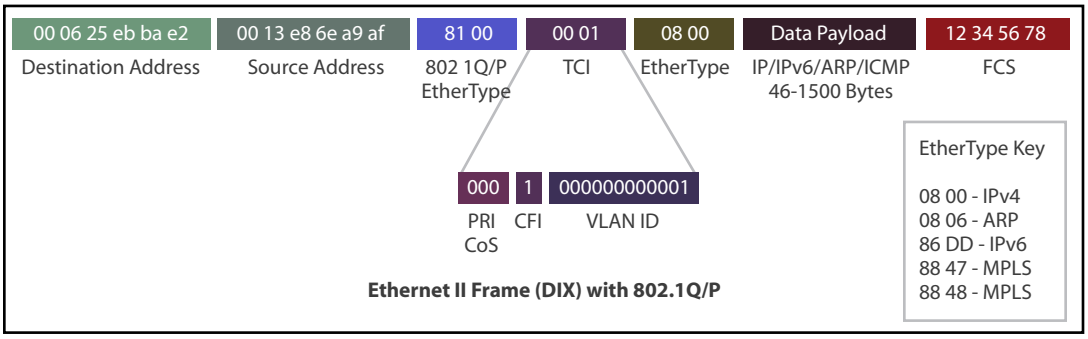


Figure 22 - IEEE 802.1Q/P Frame

Figure 23 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 23 - IEEE 802.1P CoS (PRI) Options

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 24.

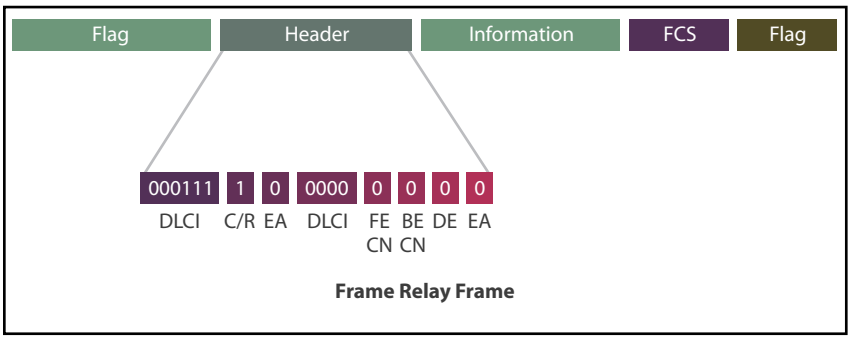


Figure 24 - Frame Relay Header

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 then 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 25.

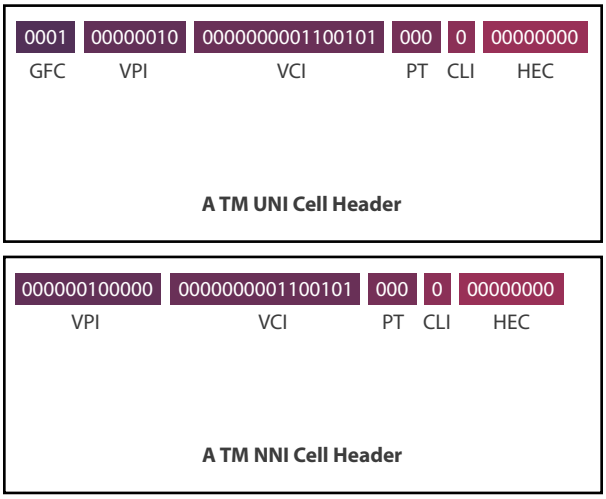


Figure 25 - ATM Cell Headers (UNI & NNI)

MPLS EXP

Unlike the previous sections, MPLS is considered a protocol which provides functionality of both Layer 2 and Layer 3 protocols, which 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 26, 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.

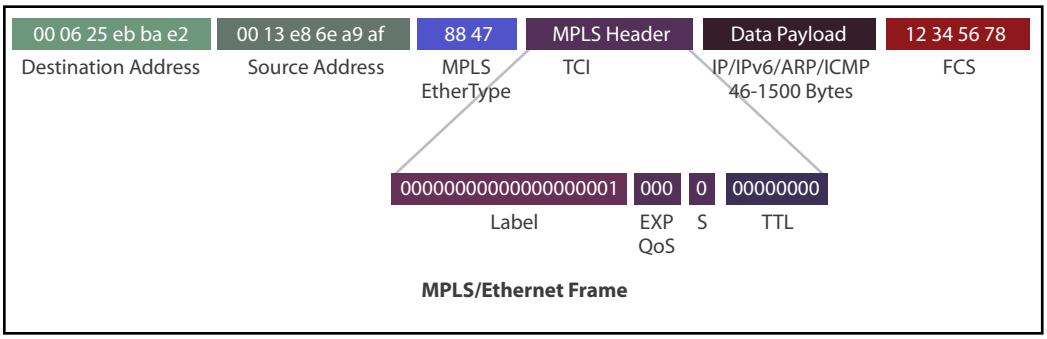


Figure 26 - MPLS Label (Ethernet Frame)

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.

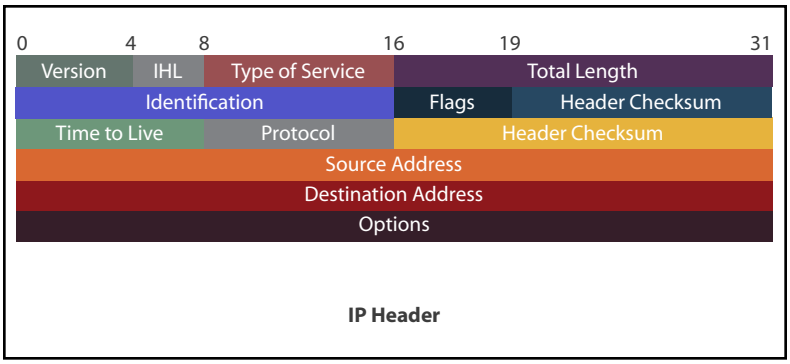


Figure 27 - IP Header

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 and the second way which 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 29.

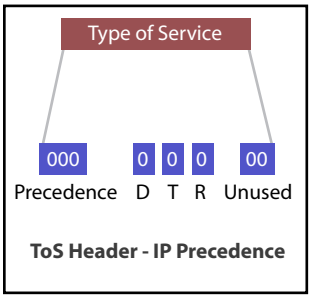


Figure 28 - IP Precedence

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

Figure 29 - 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) and Per-Hop Behaviors (PHB)

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. What DiffServ does with the ToS field is split it between a 6-bit DSCP field and a 2-bit Explicit Congestion Notification (ECN) field.

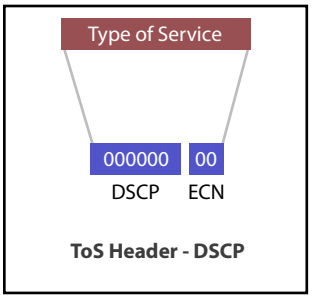


Figure 30 - DSCP

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 PHB's 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 PHB's will get preference.

The second type of PHB is the Class Selector (CS) PHB's which are designated by 'xxx000', meaning that the last three (low order) binary digits must be '000'. The Class Selector PHB's are typically used to maintain backward compatibility with IP Precedence; this is why 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 PHB's or codepoints are given a name based on the IP Precedence that they equate to, CS1 = IP Precedence 1 = '**001**000' through CS7 = IP Precedence 7 = '**111**000'. As a side note the Default PHB is also considered a Class Selector PHB ('**000**000').

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 PHB's 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 the use of 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 then 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 1 - 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 2 - 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.

Network Based Application Recognition (NBAR)

NBAR is a traffic classification engine which can be used for multiple tasks in your 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, dynamically assigned TCP/UDP ports and 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 also 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 or destined for the networking equipment performing NBAR.
Not supported on Fast Etherchannel interfaces or interfaces using tunneling or encryption.

Congestion Management and Avoidance Mechanisms

Congestion on a network is a simple concept, if the network has more traffic trying to be sent then capacity exists then congestion occurs. In order to manage congestion a number of queuing mechanisms exist that 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 we will go over these different types of congestion management and congestion avoidance mechanisms.

First In, First Out (FIFO) Queuing

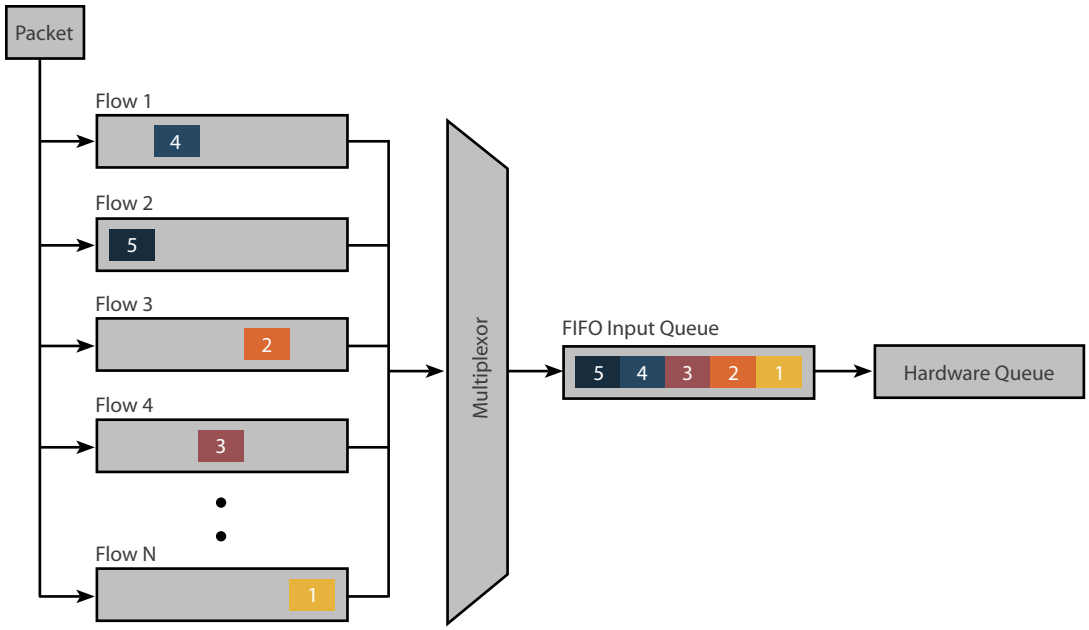


Figure 31 - FIFO Queuing Example

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 was first entered into 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)

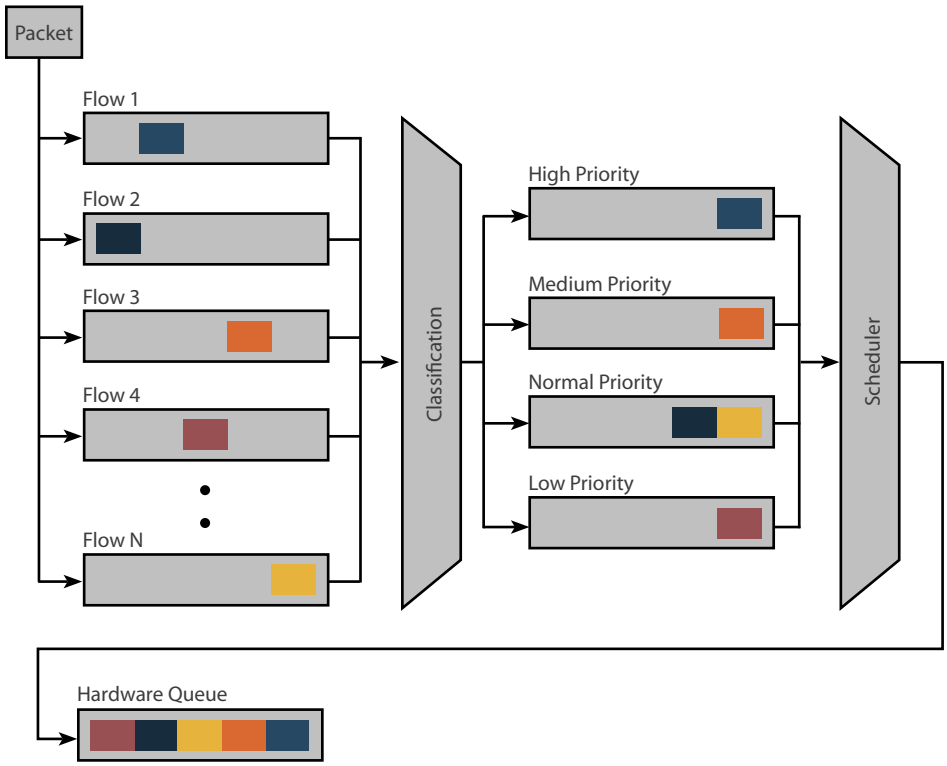


Figure 32 - Priority Queuing Example

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 are always serviced completely before the lower priority queues, this means that if there is a steady flow of higher priority traffic then the lower queues will fill and then start to tail drop packets.

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

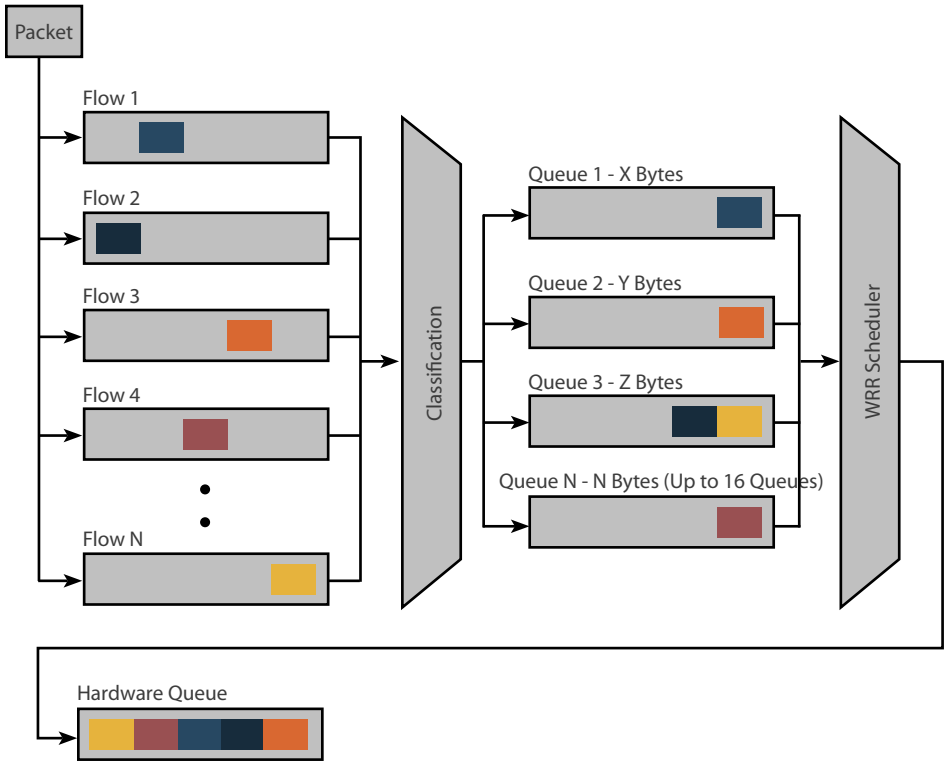


Figure 33 - Custom Queuing (WRR) Example

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 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)

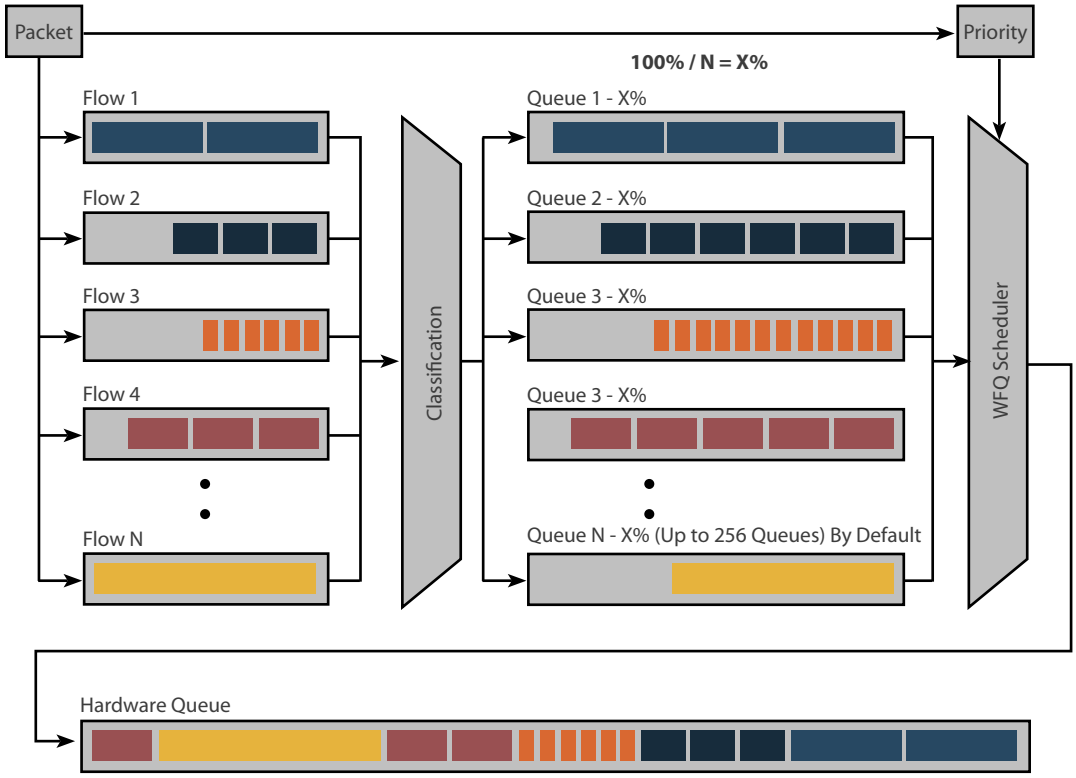


Figure 34 - Weighted Fair Queuing Example

WFQ is also called flow-based WFQ this is because of what it first does which is organizing 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 against the current queues that exist to find if there was earlier traffic with the same hash, if so the traffic is placed in the same queue 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, 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 like 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 35 - WFQ Weight Formula

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

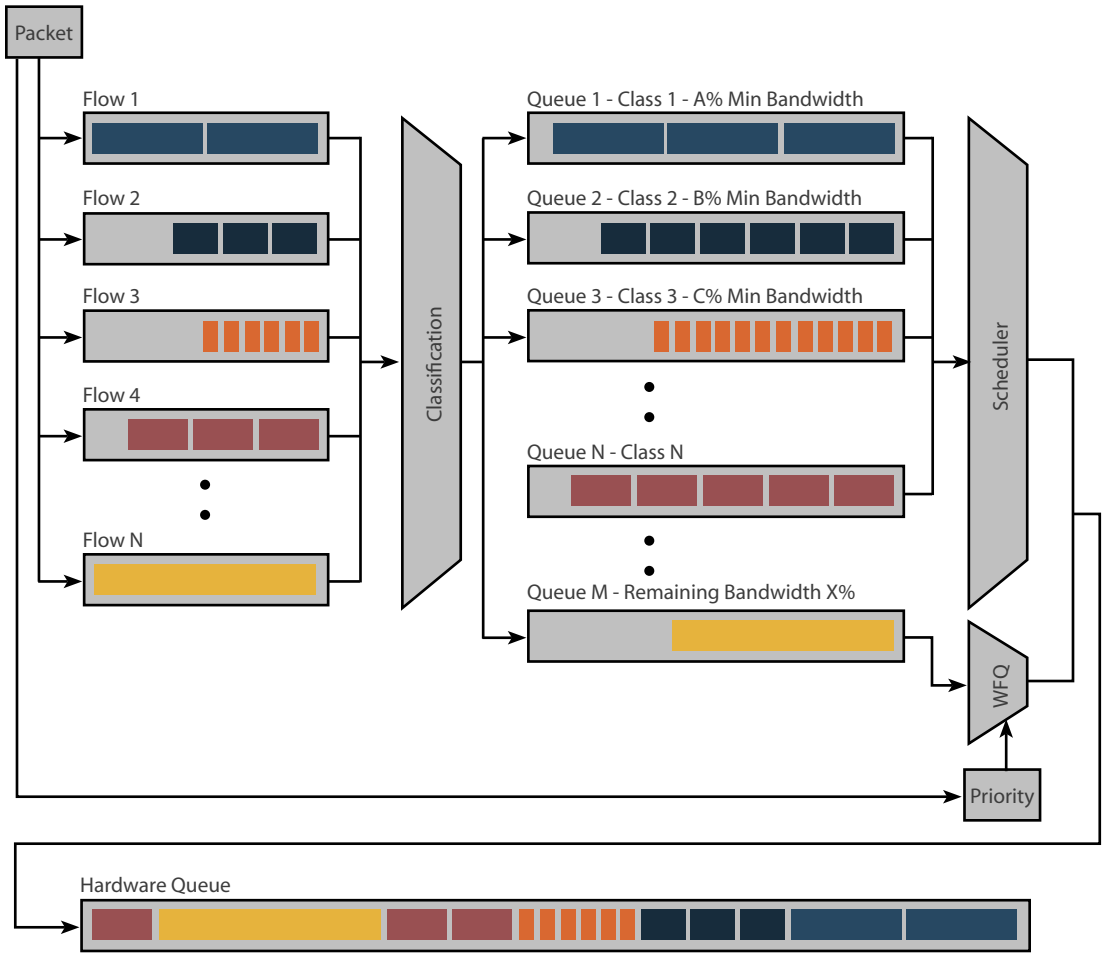
If there are three flows, one with an IP precedence of 1, one with an IP precedence of 3, and one with an IP precedence of 5, WFQ would effectively do the following with the scheduled packets:

$$1 + 3 + 5 = 8$$

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)

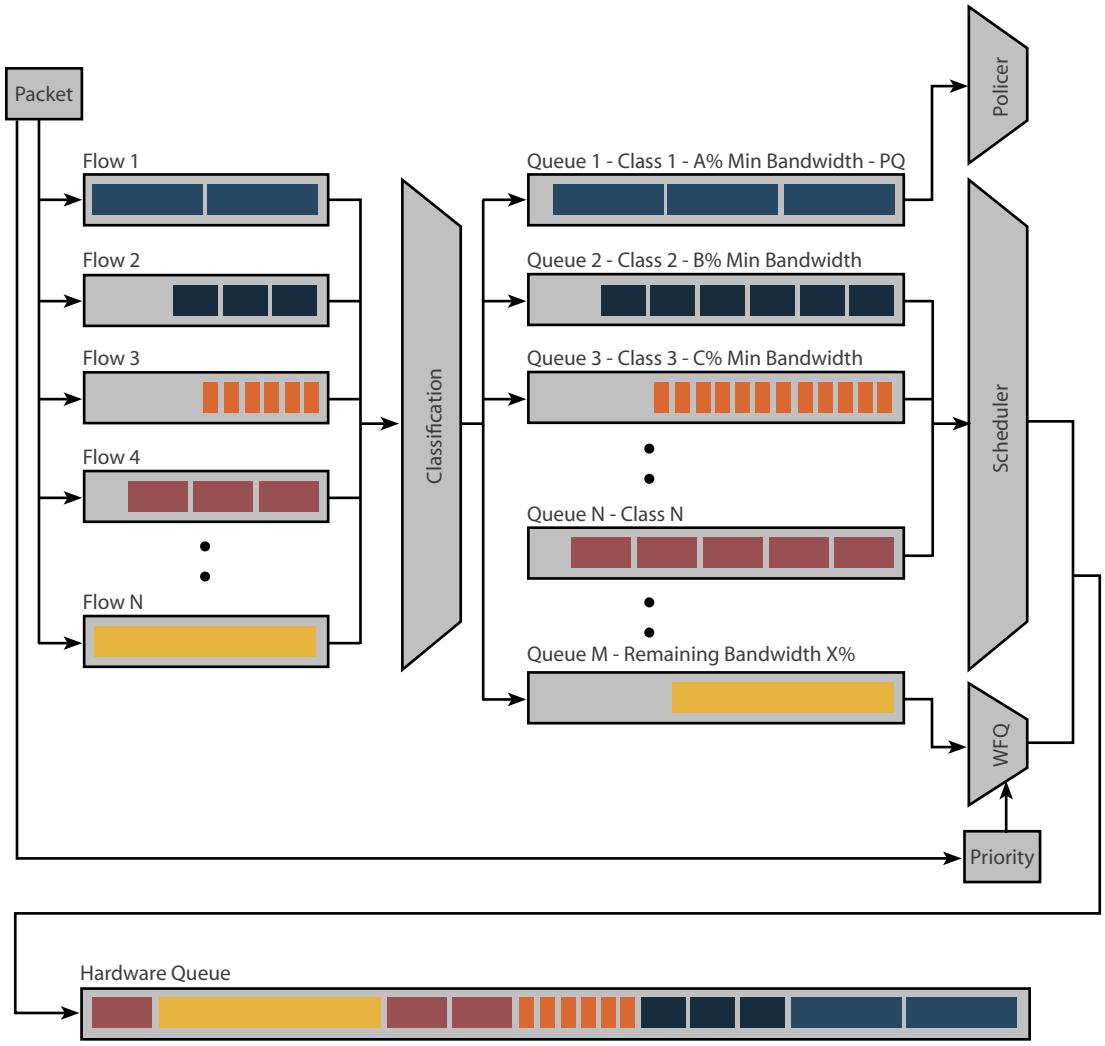


CBWFQ is a new queuing mechanism which includes a couple of advantages of other queuing mechanisms. It allows the creation of classes which can be used to set either a minimum bandwidth allowed or a minimum percentage of the interface bandwidth. This guarantee is unique to CBWFQ; CBWFQ also provides 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 then 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)



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 with 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 this queue cannot starve any of the other queues. Only the allotted amount of priority bandwidth is guaranteed in congested times.

Tail-Drop

Tail-drop is an easy concept. When a queue is filled, all additional packets which try to enter the queue are dropped until the queue empties.

Weighted Random Early Detection (WRED)

WRED is different from all the queue management options because it is geared at 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:

$$Average = (old\_average \times (1 - 2^{-n})) + (current\_queue\_size \times 2^{-n})$$

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

WRED is intended mainly to be used to control TCP traffic, this is because TCP has a built in congestion control mechanism. This mechanism is called *windowing* and is used to determine the amount of TCP packets sent before an acknowledgement. When a *window* is set to 1 then for ever 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. What WRED does is limit 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 higher.

WRED is also IP precedence, DSCP, and RSVP aware being that 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.

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 unless configured differently.

Traffic Policing & 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 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 then the traffic is sent. How excessive traffic rates and bursts are handled differ between policing and shaping.

Traffic Policing

Traffic policing handles excesses in traffic rate and burst through either 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 allows for easy configuration. CAR must first classify the traffic that is intended to be policed which 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 is able to be set, the normal burst and the excess burst. 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.

NormalBurst = *CIR* x 1 Byte / 8 Bits x 1.5
ExtendedBurst = 2 x *NormalBurst*

Figure 36 - Cisco Recommended Values

Traffic Shaping

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 37 - 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 allows for easy CLI configuration. Class-Based traffic shaping is limited to shaping outbound traffic.

Hierarchical configuration of Class-Based traffic shaping works by nesting a policy (child) inside another policy (parent), a sample of this is shown here:

```
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)#shape average 512000 (Shaping Config)
R1(config-pmap-c)#service-policy child
R1(config-pmap-c)#end
R1#
```

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 VC's are associated with the map class. Cisco equipment also supports Foresight on some of their switches this can be enabled on the map-class by using the **frame-relay adaptive-shaping foresight** command.

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.

QoS Pre-Classify

QoS Pre-Classify is used only on tunnel interfaces, virtual templates and crypto maps this is because in other circumstances it is not needed. What Pre-Classify does is provide the ability to correct 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 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. What Pre-Classify allows to be done is allow the service map to classify based on the original IP packet headers and not the new ones. In order to do this the **qos pre-classify** command must be used on the interface.

WAN Link Efficiency Mechanisms

Multilink PPP (MLP)

MLP provides the capability to fragment and interleave packets together which allows 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 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 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 the 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.

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.

Sytnax:

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 in to a specific class.

Sytnax:

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 to insert the new class-map before.

Sytnax:

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; 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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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*] commands 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* parameters is used to specify the DSCP value that is to be matched, there can be several *dscp-value* parameters defined to match multiple DSCP values.

Sytnax:

router(config-cmap)#**match** [**ip**] **dscp** *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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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 that all values above the **max** parameter are matched.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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.

Sytnax:

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

ip nbar protocol-discovery

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

Sytnax:

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

ip nbar pdlm

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

Sytnax:

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 for; the range is assigned with the *start* and *end* parameters or the *port-number* parameter.

Sytnax:

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

ip rtp priority

The **ip rtp priority** *starting-rtp-port-number port-number-range bandwidth* command is used to create a strict priority queue for RTP traffic. The *starting-rtp-port-number* parameter specifies the start of the port range to be matched. The *port-number-range* parameter specifies the end of the port range to be matched; this is done through adding the *starting-rtp-port-number* and the *port-number-range* number together to get the end of the range port number. The *bandwidth* parameter specifies the maximum allowed bandwidth in kbps, this number can be from 0 to 2000 kbps.

Sytnax:

router(config-if)#**ip rtp priority** *starting-rtp-port-number port-number-range bandwidth*

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.

Sytnax:

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 which 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.

Sytnax:

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.

Sytnax:

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

max-reserved-bandwidth

The **max-reserved-bandwidth** *percent* 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.

Sytnax:

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.

Sytnax:

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

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.

Sytnax:

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* specified 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.

Sytnax:

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* specified 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.

Sytnax:

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.

Sytnax:

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

or

router(config-pmap-c)#**random-detect exponential-weighting-constant** *exponent* rate-limit

rate-limit {**input** | **output**} {**access-group** *acl-index* | [**rate-limit access-group** *rate-limit-acl-index*] | **dscp** *dscp-value*} *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action* command is used to configure a CAR policy (Traffic Policing). The **input** parameter specifies that this policy is for the incoming interface. The **output** parameter specifies that this policy is for the outgoing interface. The **access-group** *acl*-index parameter specifies a access-list to attach this policy. The **rate-limit access-group** *rate-limit-acl-index* parameter specifies a rate-limit access-list to attach this policy. The **dscp** *dscp-value* parameter is used to specify a DSCP value that is attached to this policy. The *bps* parameter specifies the average traffic rate in bits per second; it must be in 8 kbps increments from 8000 to 2000000000. The *burst-normal* parameter specifies the normal burst time in bytes; this can be a value from 1000 to 512000000. The *burst-max* parameter specifies the excess burst size in bytes; this can be a value from 2000 to 1024000000. The **conform-action** *conform-access* is used to specify the action that will be taken should the traffic conform to the policy. The **exceed-action** *exceed-action* is used to specify the action that will be taken should the traffic exceed the policy.

Continue	Evaluate the next **rate-limit** command.
Drop	Drop the packet.
set-dscp-continue	Set the differentiated services codepoint (DSCP) (0 to 63) and evaluate the next **rate-limit** command.
set-dscp-transmit	Transmit the DSCP and transmit the packet.
set-mpls-exp-imposition-continue	Set the Multiprotocol Label Switching (MPLS) experimental bits (0 to 7) during imposition and evaluate the next **rate-limit** command.
set-mpls-exp-imposition-transmit	Set the MPLS experimental bits (0 to 7) during imposition and transmit the packet.
set-prec-continue	Set the IP precedence (0 to 7) and evaluate the next **rate-limit** command.
set-prec-transmit	Set the IP precedence (0 to 7) and transmit the packet.

Table continued on next page

set-qos-continue	Set the quality of service (QoS) group ID (1 to 99) and evaluate the next **rate-limit** command.
set-qos-transmit	Set the QoS group ID (1 to 99) and transmit the packet.
Transmit	Transmit the packet.

Table 3 - ***conform-access*** and ***exceed-action*** options

Sytnax:

router(config-if)#**rate-limit** {**input** | **output**} {**access-group** *acl-index* | [**rate-limit access-group** *rate-limit-acl-index*] | **dscp** *dscp-value*} *bps burst-normal burst-max* **conform-action** *conform-action* **exceed-action** *exceed-action*

police

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.

Drop	Drops the packet.
set-clp-transmit *value*	Sets the ATM Cell Loss Priority (CLP) bit from 0 to 1 on the ATM cell and transmits the packet with the ATM CLP bit set to 1.
set-cos-inner-transmit *value*	Sets the inner class of service field as a policing action for a bridged frame on the Enhanced FlexWAN module when using bridging features on SPAs with the Cisco 7600 SIP-200 and Cisco 7600 SIP-400 on the Cisco 7600 series router.
set-cos-transmit *value*	Sets the COS packet value and sends it.
set-discard-class-transmit	Sets the discard class attribute of a packet and transmits the packet with the new discard class setting.
set-dscp-transmit *value*	Sets the IP differentiated services code point (DSCP) value and transmits the packet with the new IP DSCP value.

Table continued on next page

set-dscp-tunnel-transmit *value*	Sets the DSCP value (0 to 63) in the tunnel header of a Layer 2 Tunnel Protocol Version 3 (L2TPv3) tunneled packet for tunnel marking and transmits the packet with the new value.
set-frde-transmit *value*	Sets the Frame Relay Discard Eligibility (DE) bit from 0 to 1 on the Frame Relay frame and transmits the packet with the DE bit set to 1.
set-mpls-experimental-imposition-transmit *value*	Sets the Multiprotocol Label Switching (MPLS) experimental (EXP) bits (0 to 7) in the imposed label headers and transmits the packet with the new MPLS EXP bit value.
set-mpls-experimental-topmost-transmit *value*	Sets the MPLS EXP field value in the topmost MPLS label header at the input and/or output interfaces.
set-prec-transmit *value*	Sets the IP precedence and transmits the packet with the new IP precedence value.
set-prec-tunnel-transmit *value*	Sets the precedence value (0 to 7) in the tunnel header of an L2TPv3 tunneled packet for tunnel marking and transmits the packet with the new value.
set-qos-transmit *value*	Sets the qos-group value and transmits the packet with the new qos-group value.
Transmit	Transmits the packet. The packet is not altered.

Table 4 - ***action*** options

Sytnax:

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

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.

Sytnax:

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

traffic-shape rate

The **traffic-shape rate** *bit-rate* [*burst-size* [*excess-burst-size*]] command is used to enable outbound GTS traffic shaping on an interface. The *bit-rate* parameter specifies the committed information rate (CIR) in bits per second; this value can be from 8000 to 100000000. The *burst-size* parameter specifies the number of burst bits (Bc) that can be sent per interval; this value can be from 0 to 100000000. The *excess-burst-size* parameter specifies the maximum number of burst bits that can be exceed the *burst-size*; this value can be from 0 to 100000000.

Sytnax:

router(config-if)#**traffic-shape rate** *bit-rate* [*burst-size* [*excess-burst-size*]]

traffic-shape group

The **traffic-shape group** *access-list bit-rate* [*burst-size* [*excess-burst-size*]] command is used to specify GTS traffic shaping on traffic that matches an access-list. The *access-list* parameter is used to specify the traffic that traffic shaping will apply to. The *bit-rate* parameter specifies the committed information rate (CIR) in bits per second; this value can be from 8000 to 100000000. The *burst-size* parameter specifies the number of burst bits (Bc) that can be sent per interval; this value can be from 0 to 100000000. The *excess-burst-size* parameter specifies the maximum number of burst bits that can be exceed the *burst-size*; this value can be from 0 to 100000000.

Sytnax:

router(config-if)#**traffic-shape group** *access-list bit-rate* [*burst-size* [*excess-burst-size*]]

traffic-shape adaptive

The **traffic-shape adaptive** *bit-rate* command is used with a frame-relay subinterface to adapt the GTS shaper based on the BECN bit of traffic. The *bit-rate* parameter is used to specify the lowest bit rate that traffic will be shaped to.

Sytnax:

router(config-subif)#**traffic-shape adaptive** *bit-rate*

traffic-shape fecn-adapt

The **traffic-shape fecn-adapt** command is used with a frame-relay subinterface when you want it to take the FECN bit from incoming traffic and translate that to the BECN bit on outgoing traffic.

Sytnax:

router(config-subif)#**traffic-shape fecn-adapt**

frame-relay traffic-shaping

The **frame-relay traffic-shaping** command is used to enable traffic shaping and per-VC queuing on a frame relay interface.

Sytnax:

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

map-class frame-relay

The **map-class frame-relay** *map-class-name* command is used to create a frame-relay map-class which defines QoS parameters. The *map-class-name* parameter specifies the name of the map-class being created or editing.

Sytnax:

router(config)#**map-class frame-relay** *map-class-name*

frame-relay adaptive-shaping

The **frame-relay adaptive-shaping** {**becn** | **foresight** | **interface-congestion** [*queue-depth*]} command is used to enable frame relay adaptive traffic shaping. The **becn** parameter is used to enable the traffic shaping response based on traffic with the BECN bit set. The **foresight** parameter is used to enable the traffic shaping response based on Foresight messages. The **interface-congestion** parameter is used to enable traffic response based on interface congestion. The *queue-depth* parameter is used in conjunction with the **interface-congestion** parameter; specifically the *queue-depth* specifies the amount of packets that are allowed to be in the interface queue before an interface is considered congested. By default, this value is 0; it can be from 0 to 40.

Sytnax:

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

frame-relay traffic-rate

The **frame-relay traffic-rate** *average* [*peak*] command is used to configure VC traffic shaping. The *average* parameter is used to specify the Committed Information Rate (CIR) in bits per second. The *peak* parameter is used to specify the peak information rate in bits per second.

Sytnax:

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

qos pre-classify

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

Sytnax:

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

ppp multilink

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

Sytnax:

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

ppp multilink interleave

The **ppp multilink interleave** command is used to enable the interleaving of packet fragments with large packets.

Sytnax:

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

ppp multilink fragment delay

The **ppp multilink fragment delay** *milliseconds* [*microseconds*] command is used to control the maximum time for transmission of a packet fragment. The *millisecond* parameter is used to set the delay in milliseconds, by default this value is 30, this value can be from 0 to 1000. The *microseconds* parameter is used to set the delay in microseconds, this parameter is optional, should the parameter be used set the *millisecond* parameter to 0.

Sytnax:

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

control-plane

The **control-plane** command is used to enter control-plan configuration mode.

Sytnax:

```
router(config)#control-plane
```

ip rtp header-compression

The **ip rtp header-compression** command is used to enable RTP header compression.

Sytnax:

```
router(config-if)#ip rtp header-compression
```

frame-relay ip rtp header-compression

The **frame-relay ip rtp header-compression** [**active** | **passive**] command is used to enable RTP header compression on frame relay maps. The **active** parameter is used to enable RTP header compression on all outgoing RTP packets. The **passive** parameter is used to enable RTP header compression only on traffic which is received with RTP header compression.

Sytnax:

router(config-if)#**frame-relay ip rtp header-compression** [**active** | **passive**]

ip tcp header-compression

The **ip tcp header-compression** [**passive** | **iphc-format | ietf-format**] command is used to enable IP TCP header compression. The **passive** parameter is used to enable IP TCP header compression only on traffic that is received with IP TCP header compression enabled. The **iphc-format** parameter specifies that the IP Header Compression (IPHC) type of header compression will be used, by default this format is used on PPP interfaces. The **ietf-format** parameter specifies that the Internet Engineering Task Force (IETF) type of header compression will be used, by default this format is used on HDLC and frame-relay interfaces.

Sytnax:

router(config-if)#**ip tcp header-compression** [**passive** | **iphc-format | ietf-format**]

compress

The **compress** {**predictor** | **stac**} command is used to enable software compression on an interface. The **predictor** parameter is used to specify the use of the predictor algorithm. The **stac** parameter is used to specify the use of the Stacker algorithm.

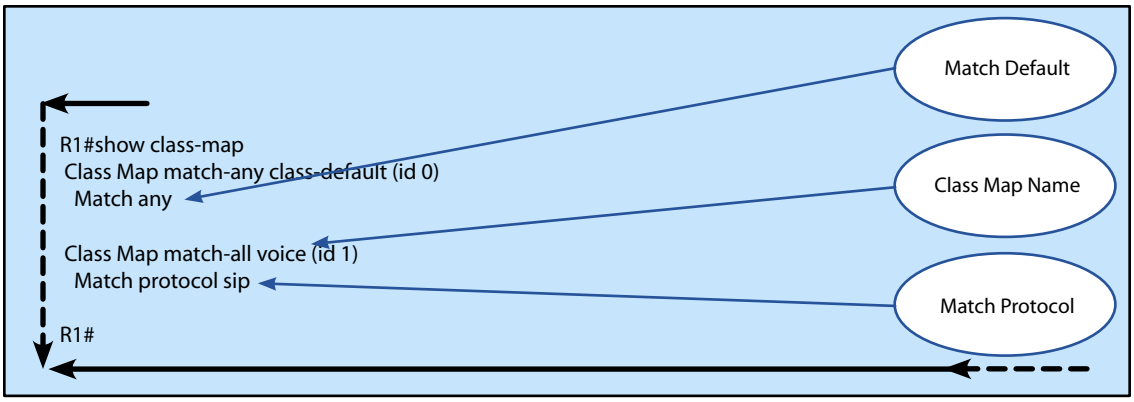
Sytnax:

router(config-if)#**compress** {**predictor** | **stac**}

Troubleshooting

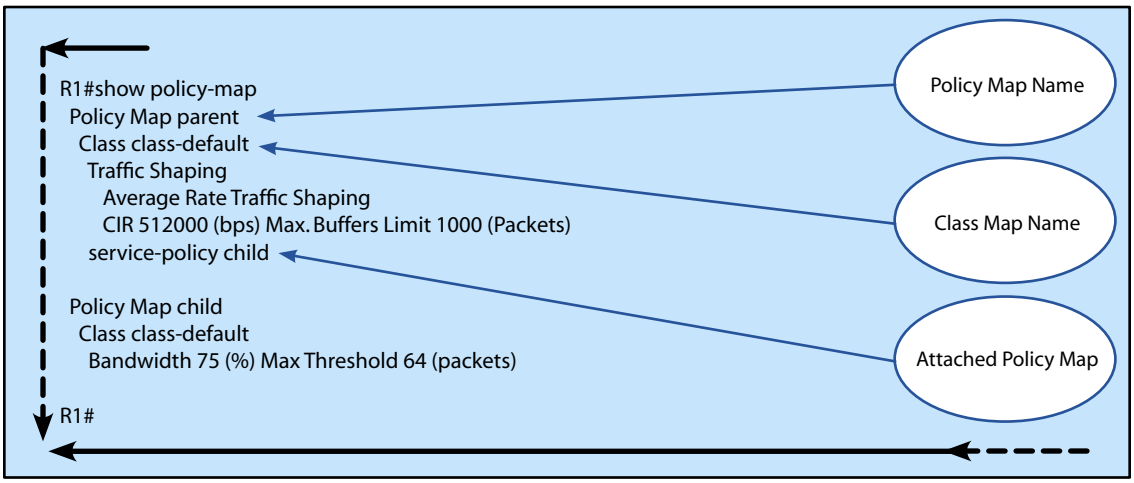
show class-map

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



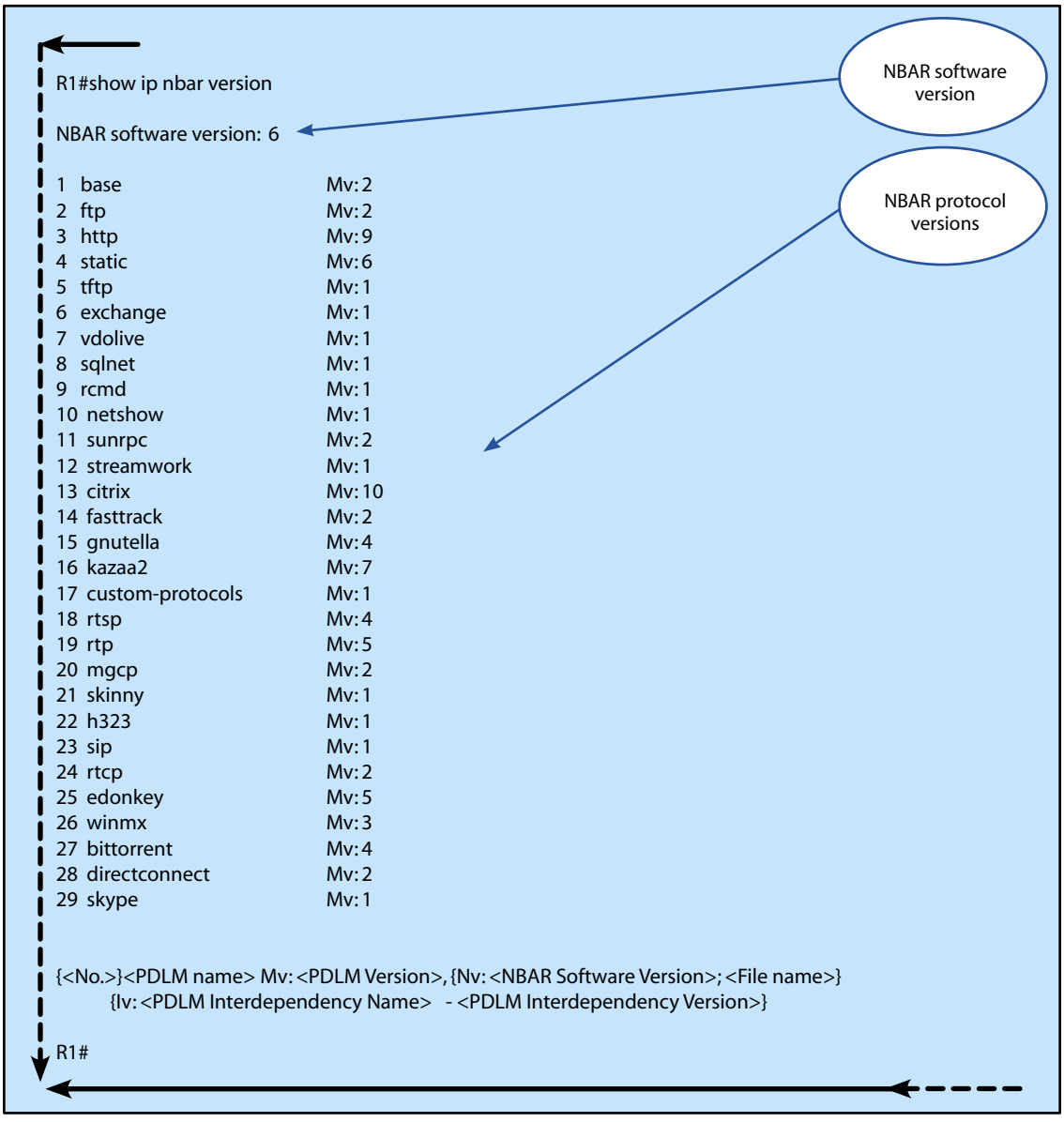
show policy-map

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



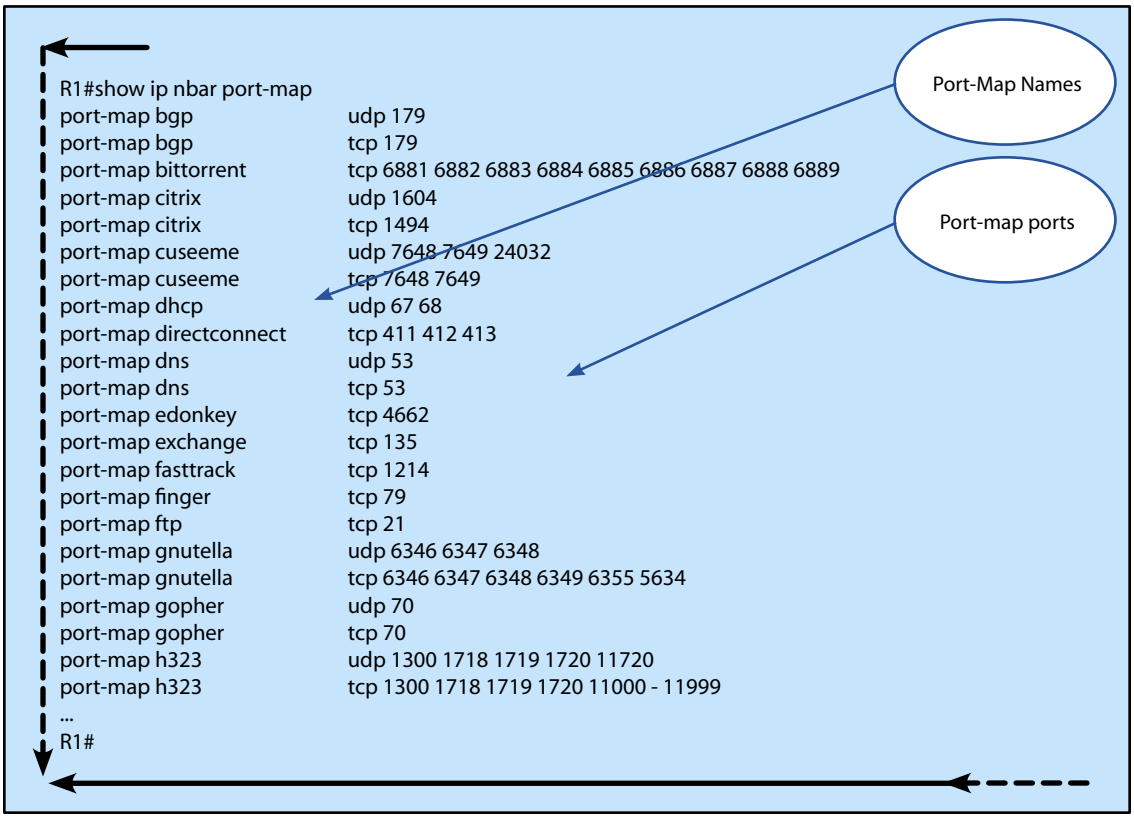
show ip nbar version

This command is used to display information about the NBAR version and NBAR protocol versions. The following highlights the most important parts.



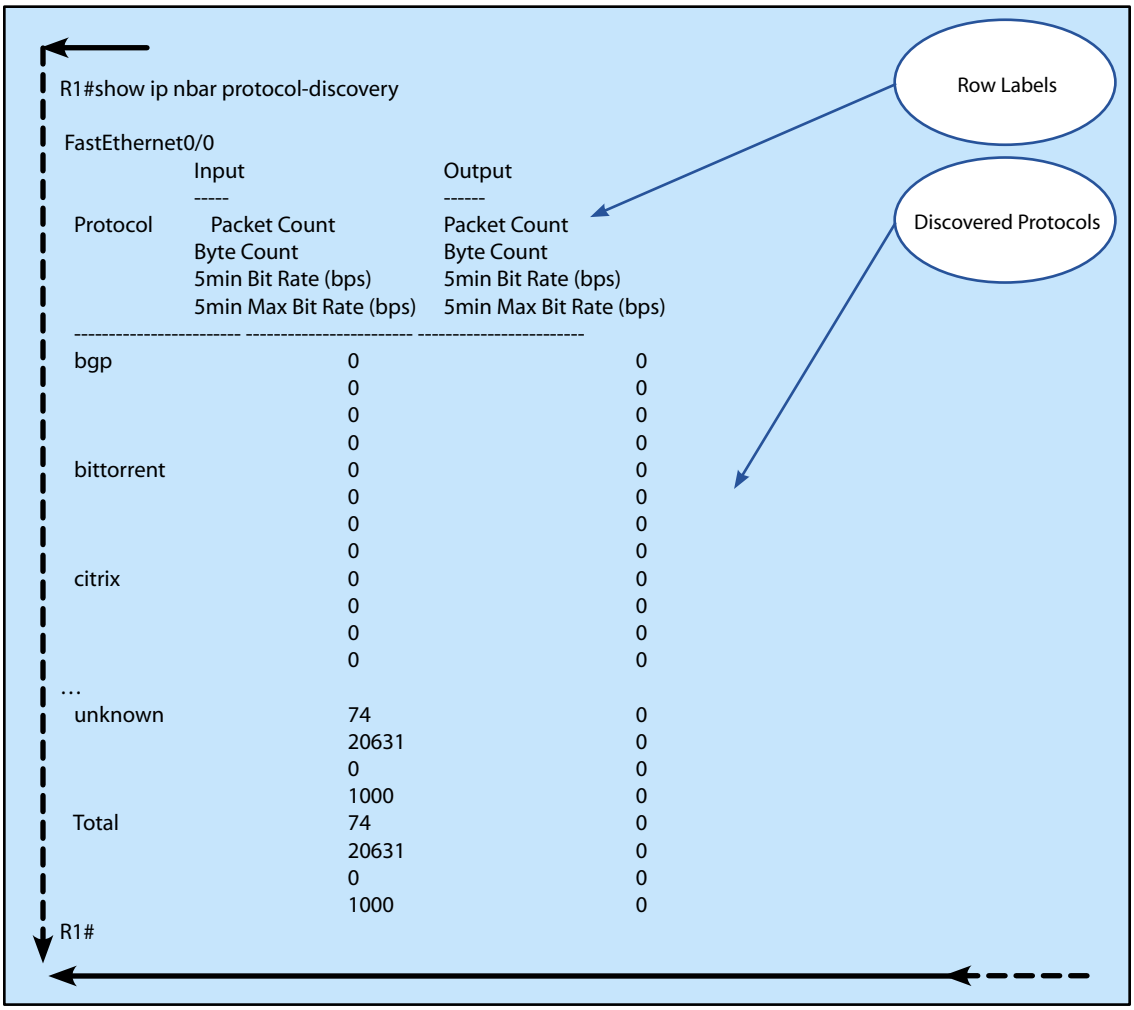
show ip nbar port-map

This command is used to display information about the ports that are mapped for NBAR discovery. The following highlights the most important parts.



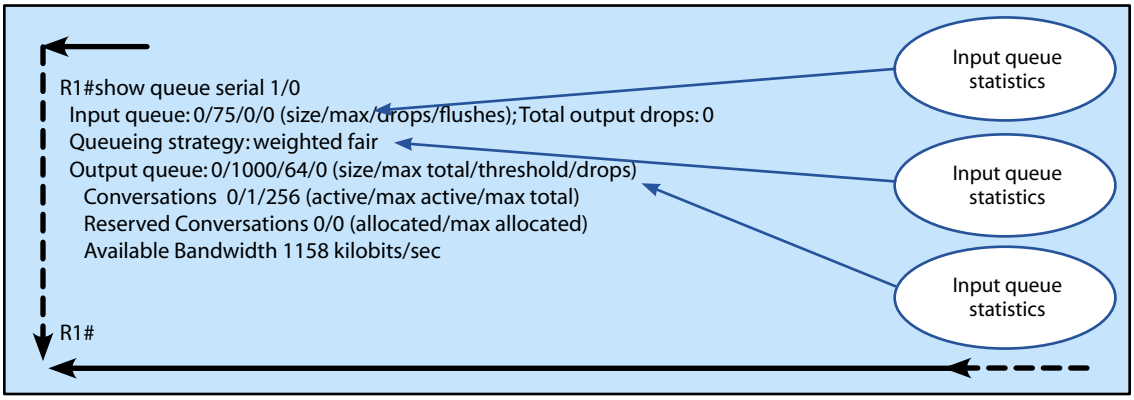
show ip nbar protocol-discovery

This command is used to display information about discovered NBAR protocols. The following highlights the most important parts.



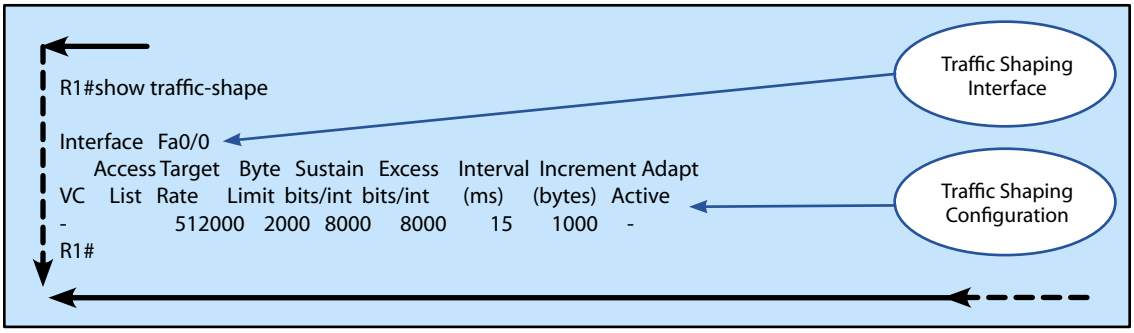
show queue

This command is used to display information about the configured queue configuration and statistics on an interface. The following highlights the most important parts.



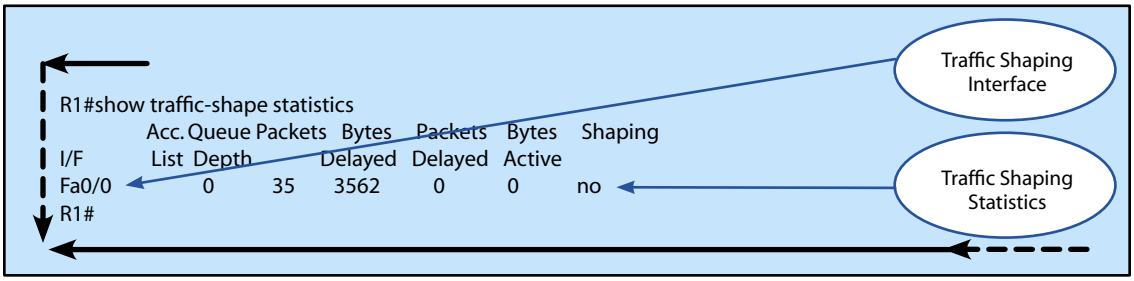
show traffic-shape

This command is used to display information about configured traffic shaping interfaces. The following highlights the most important parts.



show traffic-shape statistics

This command is used to display information about configured traffic shaping interface statistics. The following highlights the most important parts.



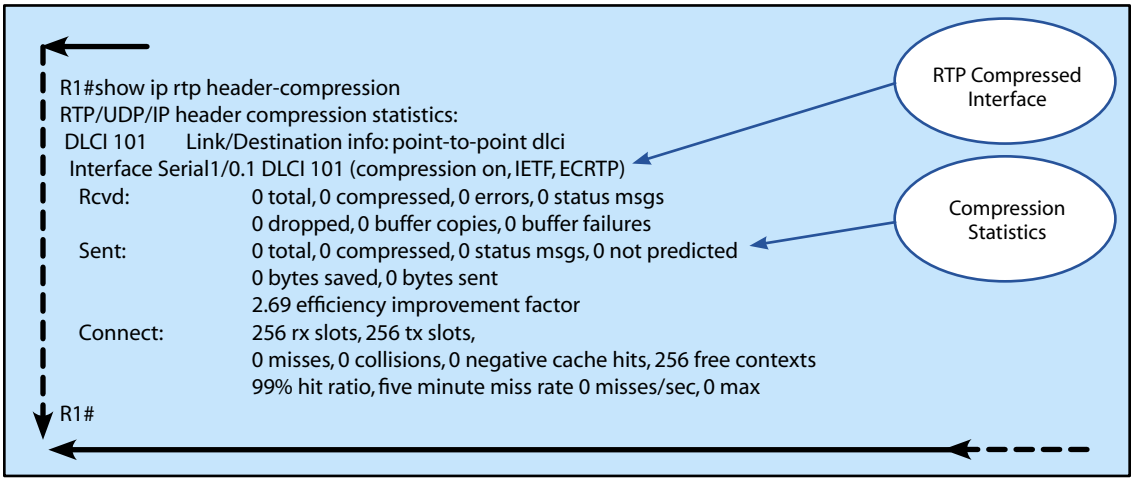
show ip tcp header-compression

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

```
R1#show ip rtp header-compression
TCP/IP header compression statistics:
 DLCI 101      Link/Destination info: point-to-point dlci
  Interface Serial1/0.1 DLCI 101 (compression on, VJ)
    Rcvd:              0 total, 0 compressed, 0 errors, 0 status msgs
                       0 dropped, 0 buffer copies, 0 buffer failures
    Sent:              0 total, 0 compressed, 0 status msgs, 0 not predicted
                       0 bytes saved, 0 bytes sent
    Connect:           256 rx slots, 256 tx slots,
                       0 misses, 0 collisions, 0 negative cache hits, 256 free contexts
R1#
```

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.



Domain 4 - Implement AutoQoS

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 automatically 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 was 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, 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 is needed for QoS configuration may be lacking. This is because of how AutoQoS works; AutoQoS generates a configuration that is 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

There are some limitations that AutoQoS in the Enterprise has, 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 DLCI's and ATM PVC's 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 tables 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 5 - 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 also 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

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 in to the traffic, this is typically used if other devices in the network are marking traffic and these devices are trusted.

Sytnax:

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.

Sytnax:

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 in to the traffic, this is typically used if other devices in the network are marking traffic and these devices are trusted.

Sytnax:

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 and requires CDP version 2 which is used by the switch to verify a Cisco phone is attached.

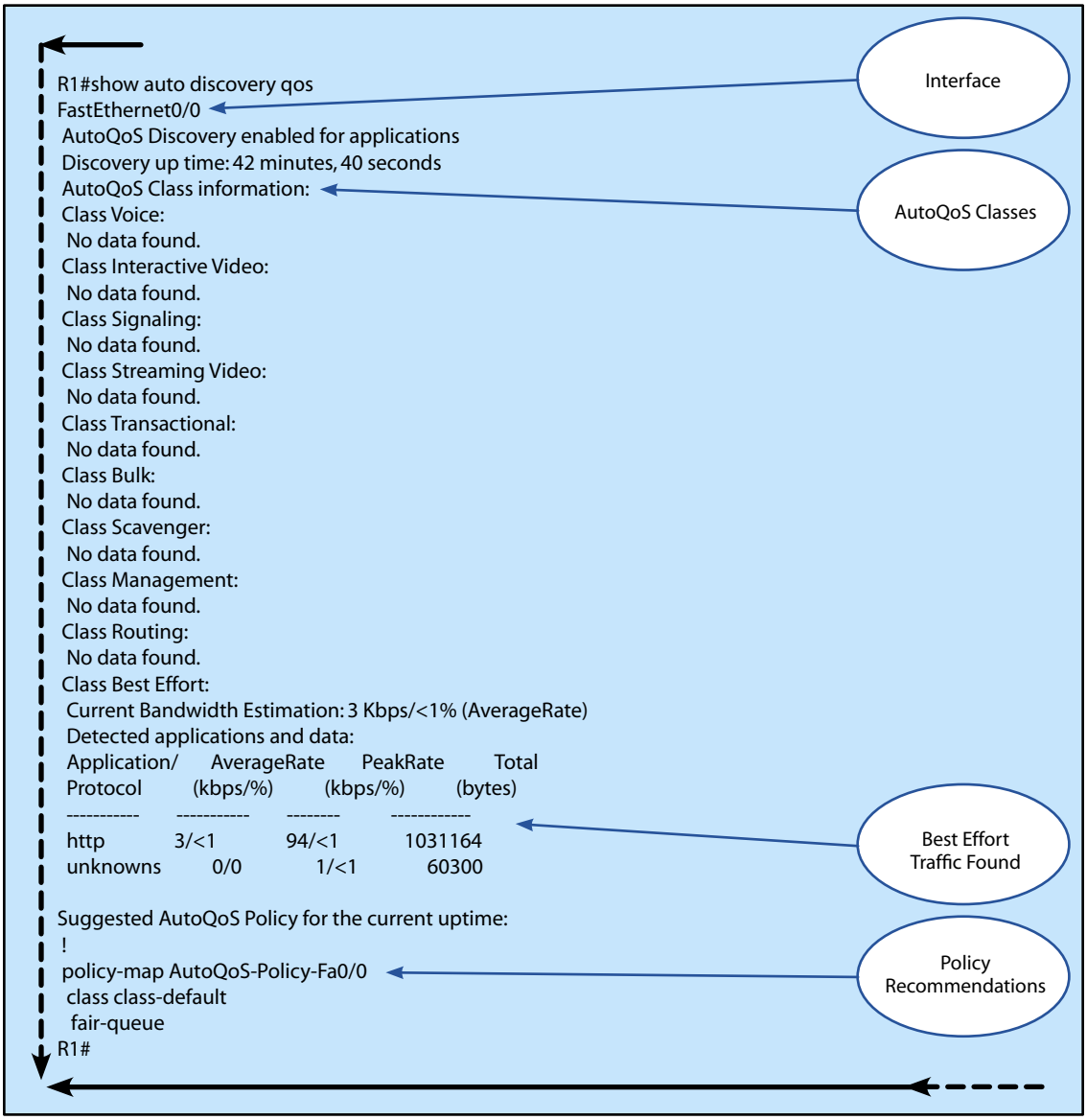
Sytnax:

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

Troubleshooting

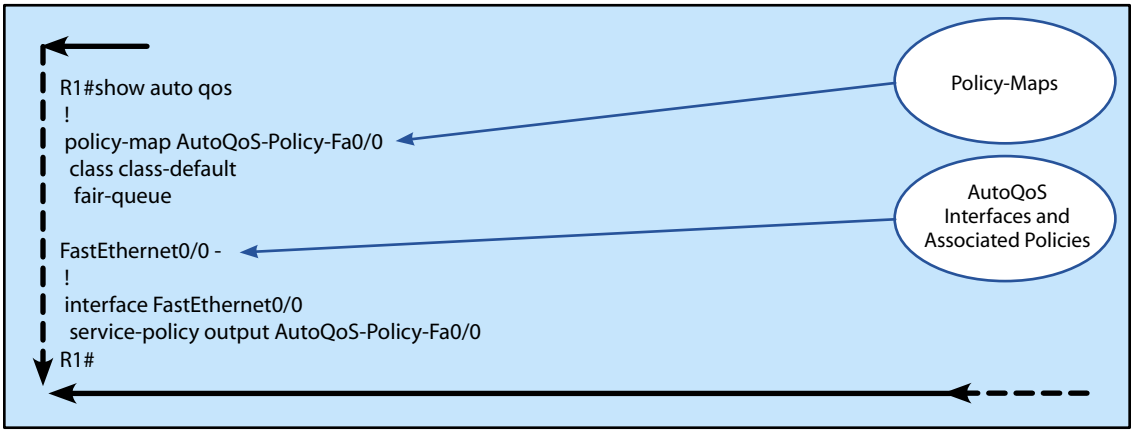
show auto discovery qos

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



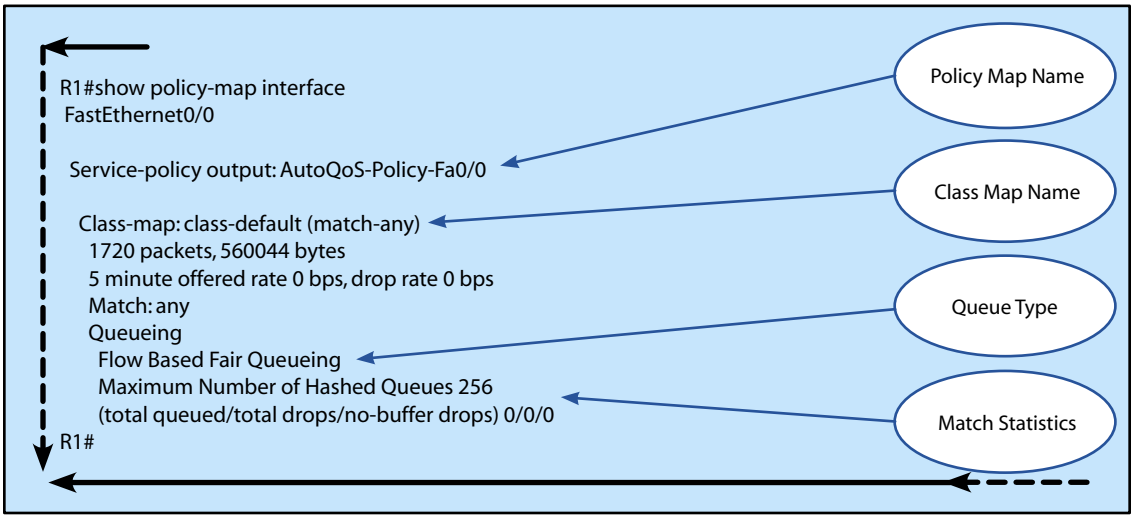
show auto qos

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 5 - Implement WLAN Security and Management

Wireless Security Basics

In order to understand wireless security several basic concepts must be understood. The first of these concepts is the *Service Set Identifier* (SSID); this is used to identify a specific wireless network. The SSID on a network can be either broadcast or not broadcast to its area; by not broadcasting the SSID, it is slightly more difficult for people to find the network if it is to be used openly. However, not broadcasting your SSID will provide little security because a client is able to obtain the SSID by simply sniffing wireless traffic.

Another concept is that of Media Access Control (MAC) filtering. This concept goes under the assumption that by limiting the MAC addresses which can speak to an Access Point (AP), you limit the exposure of the AP from rogue traffic. The problem with this concept is that with appropriate software, the spoofing of a MAC address is trivial. Therefore, this should not be considered a strong security method. It does however limit the amount of non-savvy users from connecting to the AP.

Wired Equivalent Privacy (WEP)

The early standard for wireless privacy was Wired Equivalent Privacy (WEP). WEP uses the RC4 stream cipher for confidentiality, used a 24-bit Initialization Vector (IV) and used CRC-32 for integrity. Initially, because of export restrictions, WEP used a 40-bit user key with the 24-bit IV which made a 64-bit WEP key. Once the restrictions were lifted, the WEP key grew to a 128-bit key including a 104-bit user key and the 24-bit IV. For authentication WEP provided two options; shared and open authentication. Shared authentication required WEP to be used for authentication of the client to the AP; this was accomplished through a four-way handshake which verified WEP key as authentic. Open authentication did not require authentication of the WEP keys and effectively provides no authentication, however the WEP keys are required to be valid for data encryption. The problem that was found with shared authentication was that during the four-way handshake the WEP key was able to be derived which made the security effectively useless. Open authentication is still insecure by today's standards because it can also be derived but it takes more time and captured traffic to accomplish.

Cisco in response to the weaknesses in WEP developed Lightweight Extensible Authentication Protocol (LEAP) which was later renamed the Cisco Wireless EAP. This solution provided a couple of main advantages over standard WEP. It provided for a server based authentication mechanism through passwords, tokens, certificated or machine ID's, a mechanism for using dynamic WEP keys through the use of Cisco Key Integrity Protocol (CKIP) and utilized a MMH message integrity check (MIC) which added additional protection to attacks.

WPA

In response to the weaknesses found with WEP, the WiFi Alliance brought out Wi-Fi Protected Access (WPA) which was a subset of the then proposed 802.11i standard. WPA brought many advances over WEP, including a server based authentication mechanism, a larger IV, built in dynamic keys through the use of Temporal Key Integrity Protocol (TKIP), and additional security over WEP's CRC-32 through the use of a message integrity check (MIC). WPA was configured with one of two choices, Personal or Enterprise. With WPA-Personal the network was secured through a Pre-shared key (PSK). When configured in WPA-Personal mode a user creates a passphrase which is shared between the client and the AP. This passphrase must be between 8 and 63 ASCII characters, the longer and more difficult to guess the better (similar to typical passwords). In WPA-Enterprise mode the use of 802.1x was deployed. This standard was used as a form of network access control. When using WPA-Enterprise the client talks back and forth with the AP,

but is limited to speaking to any devices other then the AP until the client and the backend authentication server agree on the clients credentials. This exchange with the backend authentication server is done through Extensible Authentication Protocol (EAP). There are various implementations of EAP that will be covered in a future section.

WPA still used the RC4 cipher for encryption but provided a longer 48-bit IV which provided additional security over the lower strength WEP 24-bit IV. WPA also used a 128-bit key like WEP but with the use of TKIP provided for dynamic keys which were derived from the passphrase and which were changed as the system was used.

WPA2

Once the 802.11i standard was approved, it was named WPA2. WPA2 like WPA provided for two modes of operation, Personal and Enterprise which work in the same way as WPA. The primary difference between WPA and WPA2 are the use of AES-Based Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) for authentication, confidentiality and integrity. AES is a block cipher unlike the RC4 stream cipher and is considered to be a considerably higher strength encryption mechanism. Like WPA, WPA2 utilizes 802.1x for server based authentication in Enterprise mode and a PSK in Personal mode.

One of the main disadvantages of WPA2 is that it requires a larger processer compared with WPA because of the use of AES. Because of this many pieces of old equipment are not able to conform to the WPA2 standard without hardware upgrade while the WPA standard was able to be achieved typically without a hardware upgrade.

802.1x and Extensible Authentication Protocol (EAP)

802.1x Overview

802.1x is a standard that is built around implementing Network Access Control (NAC). 802.1x defines three different duties, the supplicant, the authenticator, and the Authentication Server (AS) which is typically a Remote Authentication Dial In User Service (RADIUS) server. The supplicant is run on the client; it allows the client to speak with the authenticator and the AS. The authenticator in the wireless world is run on the AP. The authenticator speaks with the client and allows the EAP traffic to pass-through to the AS. The EAP is what is used for the authentication between the client and the AS. When using 802.1x the client is able to talk back and forth from the AS with EAP messages but all other traffic is limited between the client and the AP. Once the client is able to authenticate with the AS then all traffic is allowed.

Lightweight Extensible Authentication Protocol (LEAP)

LEAP was developed by Cisco in the time between WEP and WPA. It enabled the use of 802.1x to authenticate a client with a username and password. This authentication was done using the Cisco Key Integrity Protocol (CKIP) and the authentication database was a RADIUS server. This type of EAP is generally considered to be deprecated because of vulnerabilities, if LEAP is the only option available, it is recommended that it be used with a highly complex password only.

Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling (EAP-FAST)

EAP-FAST was developed by Cisco in order to be a replacement for LEAP. EAP-FAST provides security by setting up a session between the client and the AS via TLS. EAP-FAST for this process defined three Phases. Phase 0 is when the client dynamically provisions a Protected Access Credential (PAC); this PAC is then used for Phase 1. In Phase 1 the client and the RADIUS server setup a mutually authenticated secure tunnel and Phase 2 is used to establish the tunnel. Authentication with EAP-FAST is done with a username, password and the use of the PAC.

Extensible Authentication Protocol-Transport Layer Security (EAP-TLS)

EAP-TLS is IETF standard and is considered to be one of the most secure EAP implementations. EAP-TLS makes use of Public Key Infrastructure (PKI); this provides a TLS certificate for the server and for the client which is required. This means that in order to authenticate to the RADIUS server the client must have a username and password combination that match and the correct TLS private key.

Protected Extensible Authentication Protocol (PEAP) (EAP-MSCHAP V2)

PEAP is based on EAP-TLS, but this type instead of using a client certificate uses Microsoft Challenge-Handshake Authentication Protocol (MSCHAP) V2 for client authentication to the RADIUS server. Server authentication with PEAP uses a TLS certificate. PEAP uses a dynamic session-based WEP keys.

Protected Extensible Authentication Protocol (PEAP) (EAP-GTC)

This type of PEAP uses Generic Token Card (GTC) for authentication this is done typically through a one-time token but can also be done through databases like Lightweight Directory Access Protocol (LDAP).

Wireless LAN Management

Within Cisco wireless LAN management there are five elements, client devices, mobility platform, network unification, network management, and unified advanced services. The client devices can be a variety of devices which attach to the WLAN. The mobility platform includes lightweight access points (LWAP) and bridges. Network unification includes the Wireless LAN Controllers (WLC) including the 2000 and 4400 series WLC as well as the other Cisco devices which include WLC cards. Within Cisco wireless there are two paths for deployment, the use of autonomous access points and the use of LWAP's and WLC's.

Autonomous AP's and WLSE

When using autonomous access points, each AP runs its own configuration which is configured per AP. Autonomous AP's also use a different management system called the Wireless LAN Solution Engine (WLSE) and Wireless Domain Services (WDS). WLSE supports HTTP, SSH, SNMP and CDP for management and provides for configuration management, policy and fault management, reporting capabilities, firmware control and radio management with radio monitoring done through WDS. WLSE is intended for medium and large businesses with support for up to 2500 managed devices. There is also an express version of WLSE for use with smaller businesses with support for up to 100 WLAN devices. Both WLSE and WLSE express have Intrusion Detection System (IDS) capabilities; this enables WLSE to detect rogue AP's and ad hoc networking equipment. WLSE also gives the ability to perform automatic surveys to map out WLAN equipment and provide for automatic healing of the WLAN. Auto healing is performed through AP power adjustment to cover the area of down AP's. WLSE express also has a built in Authentication, Authorization and Accounting (AAA) server which does not exist with the non-express edition. WLSE Express also has the ability to be configured automatically through DHCP and TFTP.

Lightweight Access Points (LWAP) and Wireless Control System (WCS)

If LWAP's and WLC's are deployed then using a different management model and software. With this type of implementation the Wireless Control System (WCS) is used. With this type of deployment the LWAP's are essentially dumb and are controlled through the WLC's. Management can be done either through the WLC's or through the WCS central management system. The WCS has the capability to support up to 50 WLC's and up to 1500 LWAP's.

WCC has three main versions; this includes WCC Base, WCC Location and WCC Location + 2700 Series Wireless Location Appliance.

WCC Base includes the following features:

Autodiscovery of access points as they associate with controllers
Autodiscovery and containment or notification of rogue access points
Map-based organization of access point coverage areas, which is helpful when the enterprise spans more than one geographical area
Detection of Rogue ad-hoc networks
User-supplied campus, building, and floor plan graphics, which show the following:
-Locations and status of managed access points on top of user supplied graphics
-Locations of rogue access points based on the signal strength received by the nearest managed Cisco access points on top of user supplied graphics
-Coverage hole alarm information for access points based on the received signal strength from Clients on top of user supplied graphics. This information appears in a tabular rather than map format.
-RF coverage maps

System-wide control of the following:
Streamlined network, controller, and managed access point configuration using customer-defined templates
Network, controller, and managed access point status and alarm monitoring
Automated and manual data client monitoring and control functions
Automated monitoring of rogue access points, rogue ad hocs, coverage holes, security violations, controllers, and access points
Full event logs for data clients, rogue access points, coverage holes, security violations, controllers, and access points
Automatic channel and power level assignment by radio resource management (RRM)
User-defined automatic controller status audits, missed trap polling, configuration backups, and policy cleanups
Real-time location of rogue access points and rogue ad hocs to the nearest Cisco access point
Real-time and historical location of clients to the nearest Cisco access point

The following are the features added with the WCC Location version:

On-demand location of rogue access points and rogue ad hocs to within 33 feet (10 meters).
On-demand location of clients to within 33 feet (10 meters).
Ability to use location appliances to collect and return historical location data viewable in the WCS Location user interface.

The 2700 Series Wireless Location Appliance provides all the functionality of the WCC Location version and adds the ability to track 1500 WLAN devices simultaneously, whereas the version without the appliance can only track one at a time.

Wireless LAN Quality of Service

The issues with Quality of Service in a wireless LAN environment are the same as with wired networks. Some types of traffic require different traffic characteristics then others; because of this, different QoS mechanisms have been created. Specific to wireless LAN's there are two main QoS methods, Wi-Fi Multimedia (WMM) and 802.11e. WMM is simply a subset of 802.11e and was brought out before 802.11e was approved.

There first needs to be a simple overview of how wireless networks work without QoS. Without QoS a wireless network works by forwarding traffic with the best effort mechanism; meaning that traffic that is received first is sent first as the equipment can manage. 802.11 specifically works through the use of a Carrier Sense – Multiple Access – Collision Avoidance (CSMA/CA), this means is that unlike a wired LAN which works through the use of collision detention a wireless LAN works by using collision avoidance. This mechanism of collision with 802.11 is called Distributed Coordination Function (DCF). Within DCF there are two main terms that must be understood, the Contention Window (CW) and the DCF Interframe Space (DIFS). The CW is also called a backoff timer or the *random wait timer*,; this is used within the CA mechanism and is calculated by multiplying a random time by the slot time (the slot time depends on the type of wireless (a/b/g/n) – example; with 802.11a it is 9*us*). The DIFS is used after the channel is detected as available, when it becomes available it must wait for the DIFS timer to expire (34*us* with 802.11a) then it must wait for the CW timer to expire. If the channel remains clear by the end of these timers then the traffic is sent; if the traffic becomes busy then the process must be started over.

With WMM and 802.11e uses Enhanced Distributed Coordination Function (EDCF). EDCF provides QoS through the use of different CW values for each priority level; the higher the priority of the traffic, the smaller the CW value.

WMM splits this traffic into four access categories and 802.11e splits them into eight priority levels, these are mapped together like the following:

WMM	802.11e
Voice (Platinum)	6 or 7
Video (Gold)	4 or 5
Best-Effort (Silver)	0 or 3
Background (Bronze)	1 or 2

Table 6 - WMM/802.11e Mappings

Wired to Wireless Connections

In order for the wireless LAN priority to be continued along the wired path a translation from Wireless QoS to Wired QoS must be done. With autonomous AP's this is done through an 802.1q port from the AP to the switch. When using LWAP's and WLC's, Cisco uses the Lightweight Access Point Protocol (LWAPP) to make sure that the QoS information from the wireless LAN to the LAN and from the LAN to the Wireless LAN are is kept intact. LWAPP also provides for the ability to run either over a Layer 2 Ethernet frame or over a Layer 3 UDP/IP packet. These provide the ability for the wireless QoS information to be translated into either 802.1p (CoS) Layer 2 or IP Precedence/DSCP Layer 3. The translation between WMM/802.11e and 802.1p or IP Precedence/DSCP is as follows:

IP DSCP	802.1p Priority	802.11e Priority
56–62	7	7
48	6	7
46 (EF)	5	6
34 (AF41)	4	5
26 (AF31)	3	4
18 (AF21)	2	2
10 (AF11)	1	1
0 (BE)	0	0 or 3

Split MAC Architecture

In order to have a central WLAN management solution, Cisco created the split MAC architecture. The split MAC architecture takes the duties that are all given to an autonomous AP and splits them into two main categories, Real-Time and Non-Real-Time duties. These duties are detailed as follows:

Beacon Generation
Probe Transmission and Response
Power Management
802.11e/WMM scheduling and queuing
MAC layer data encryption/decryption
Control Frame/message processing
Packet buffering

Table 7 - MAC Real-Time Functions

Association/Re-Association /Disassociation
802.11e/WMM Resource Reservation
802.1x EAP
Key Management
Authentication
Fragmentation
Bridging between Ethernet and WLAN

Table 8 - Non-Real-Time Functions

Within the split MAC architecture the Real-Time functionality is handled by the LWAP and the Non-Real-Time functionality is handled by the WLC.