CHAPTER 3

Medianet WAN Aggregation QoS Design 4.0

Overview

The case for Quality of Service over the enterprise medianet WAN/VPN is largely self-evident, as these links are often orders of magnitude slower than the (Gigabit or Ten-Gigabit Ethernet) campus or branch LAN links they connect to. As such, these WAN/VPN edges usually represent the greatest bottlenecks in the network and therefore require the most attention to QoS design.

Two key strategic QoS design principles, presented in Chapter 1, "Enterprise Medianet Quality of Service Design 4.0—Overview" which are highly applicable to WAN/VPN QoS design include:

- Enable queuing policies at every node where the potential for congestion exists, which generally equates to attaching a comprehensive queuing policy to every WAN/VPN edge.
- Protect the control plane and data plane by enabling control plane policing (on platforms supporting this feature) as well as data plane policing (scavenger class QoS) to mitigate and constrain network attacks.

To this end, this design chapter provides best practice recommendations for enabling QoS over the wide area network. However, it is important to note that the recommendations in this chapter are not autonomous, but rather, these depend on the campus QoS design recommendations, presented in Chapter 2, "Medianet Campus QoS Design 4.0," having already been implemented. Traffic traversing the WAN can thus be safely assumed to be correctly classified and marked with Layer 3 DiffServ codepoints (as well as policed at the access-edge, as necessary).

Furthermore, this design chapter will cover fundamental considerations and designs relating to wide area networks, as well as specific recommendations for Layer 2 private WANS; whereas subsequent chapters will cover additional considerations and designs specific to Layer 2 and Layer 3 VPNs, including MPLS VPNs, Metro Ethernet VPNs and IPSec-based VPNs.

Before strategic QoS designs for the WAN can be arrived at, a few WAN-specific considerations need to be taken into account, as are discussed below.

Medianet WAN QoS Design Considerations

There are several considerations that factor into WAN and VPN QoS designs, including:

- Medianet WAN Aggregation Router Platforms
- Hardware versus Software QoS
- Latency and Jitter
- Tx-Ring

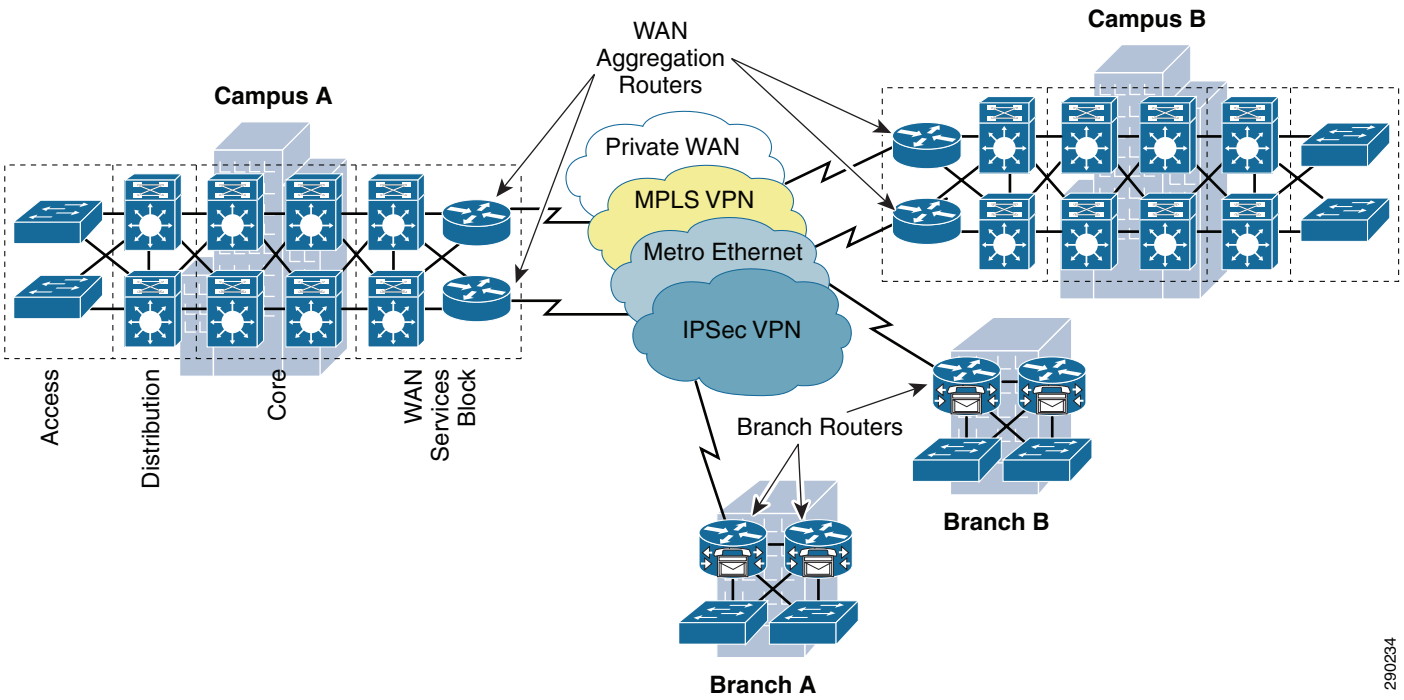
- Class-Based Weighted-Fair Queuing
- Low-Latency Queuing
- Weighted-Random Early Detect
- Link Types and Speeds
- PAK_Priority
- Resource Reservations
- Medianet WAN Interface Roles
- AutoQoS
- Control Plane Policing

Each of these WAN QoS design considerations is discussed in the following sections.

Medianet WAN Aggregation Router Platforms

Extending an enterprise campus network over a wide area to interconnect with other campus and/or branch networks usually requires two types of routers to be deployed: WAN aggregation routers and branch routers. WAN aggregation routers serve to connect large campus networks to the WAN/VPN, whereas branch routers serve to connect smaller branch LANs to the WAN/VPN. These WAN two main types of WAN routers are illustrated in Figure 3-1.

Figure 3-1 Medianet WAN and Branch Routers



WAN aggregation routers recommended for medianet deployments include:

- Cisco 7200VXR NPE-G2 routers

- Catalyst 6500/7600 routers with Shared Port Adapter Interface Processors (SIPs) and Shared Port Adapters (SPAs)
- Cisco Aggregation Services Router s (ASR) 1000 routers with Embedded Services Processors (ESPs).

On the other hand, medianet branch routers (which will be discussed in detail in a subsequent design chapter) include the Integrated Services Router G2 family of Cisco 1900, 2900 and 3900 series routers.

Hardware versus Software QoS

Unlike Cisco Catalyst switches utilized within the medianet campus, which perform QoS exclusively in hardware, Cisco routers perform QoS operations in IOS software, although some platforms (such as the Cisco Catalyst 6500/7600 and Cisco ASRs) perform QoS in a hybrid-mix of software and hardware.

Performing QoS in IOS software allows for several advantages, including:

- Cross-platform consistency in QoS features—For example, rather than having hardware-specific queuing structures on a per-platform or per-linecard basis (as is the case for Cisco Catalyst switches), standard software queuing features, like Low-Latency Queuing (LLQ) and Class-Based Weighted-Fair Queuing (CBWFQ) can be utilized across WAN and branch router platforms.
- Consistent QoS configuration syntax—The configuration syntax for IOS QoS, namely the Modular QoS Command Line Interface (MQC) syntax is-with very few exceptions-identical across these WAN and branch router platforms.
- Richer QoS features—Many IOS QoS features, such as Network Based Application Recognition (NBAR) and Hierarchical QoS (HQoS) are not available on most Catalyst hardware platforms.

However, despite these advantages, there is one main caveat to keep in mind with respect to software QoS policies, namely that these require a marginal CPU load to process. How much of an incremental CPU load each QoS policy will require will depend on many factors, including: the router platform (including any hardware-acceleration features), link speeds, policy-complexity, traffic mix and packet rates/sizes.

Table 3-1 provides a scalability guideline for the WAN aggregation routers discussed in this design chapter. However, it cannot be overemphasized that there is no substitute for testing to ensure that a given router will fit the required role. Cisco recommends that WAN or branch routers not consume more than 75% of their CPU resources during normal operation (including all QoS policy operations), in order that the CPU always have cycles available to efficiently respond to network events. If a given WAN aggregation or branch router exceeds this 75% CPU utilization target during normal operation, then it may be advisable to consider a platform upgrade.

Table 3-1 Medianet WAN Aggregation Routers Performance Capacities

WAN Aggregation Router Platform	Performance Capacity
Cisco 7200VXR with NPE-G2	1.8 Gbps
Cisco Catalyst 6500 / 7600 with SIP-200	1 Gbps
Cisco Catalyst 6500 / 7600 with SIP-400	4 Gbps
Cisco Catalyst 6500 / 7600 with SIP-600	10 Gbps
Cisco ASR 1002 with ESP 2.5, ESP5 or ESP10	2.5 Gbps, 5 Gbps or 10 Gbps, respectively
Cisco ASR 1004 with ESP10 or ESP20	10 Gbps or 20 Gbps, respectively
Cisco ASR 1006 with dual ESP10 or dual ESP20	20 Gbps or 40 Gbps, respectively

However, as mentioned, some of these WAN aggregation routers perform some QoS operations in hardware. Therefore, it's important to understand the operation of QoS processes (both hardware and software) on these platforms, as well as any platform-specific configuration syntax and caveats. Therefore, in the design section of this chapter platform-specific consideration will be given to both the Cisco Catalyst 6500/7600 and the Cisco ASRs.

Latency and Jitter

Some realtime applications have fixed latency budgets; for example, the ITU G.114 specification sets the target for one-way latency for realtime voice/video conversations to be 150 ms. In order to meet such targets, it is important for administrators to understand the components of network latency so they know which factors they can and cannot control with network and QoS design. Network latency can be broken down into fixed and variable components:

- Serialization (fixed)
- Propagation (fixed)
- Queuing (variable)

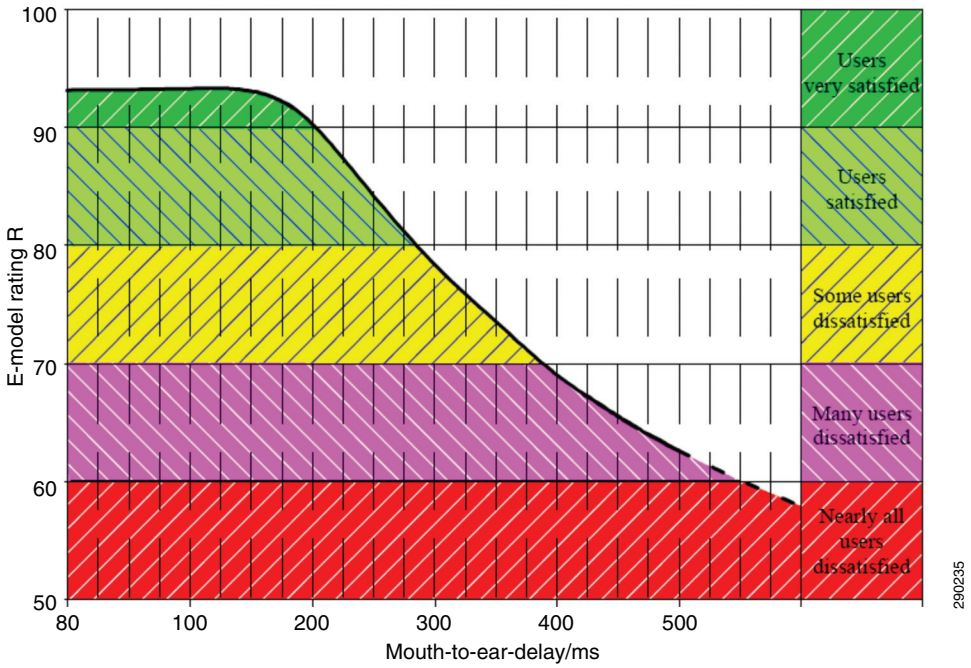
Serialization refers to the time it takes to convert a Layer 2 frame into Layer 1 electrical or optical pulses onto the transmission media. Therefore, serialization delay is fixed and is a function of the line rate (i.e., the clock speed of the link). For example, a (1.544 Mbps) T1 circuit would require about 8 ms to serialize a 1500 byte Ethernet frame onto the wire, whereas a (9.953 Gbps) OC-192/STM-64 circuit would require just 1.2 µs to serialize the same frame.

Usually, the most significant network factor in meeting the latency targets for over the WAN is propagation delay, which can account for over 95% of the network latency time budget. Propagation delay is also a fixed component and is a function of the physical distance that the signals have to travel between the originating endpoint and the receiving endpoint. The gating factor for propagation delay is the speed of light: which is 300,000 km/s or 186,000 miles per second, in a vacuum. However, the speed of light in an optical fiber is about one third the speed of light in a vacuum. Thus, the propagation delay for most fiber circuits works out to be approximately 6.3 µs per km or 8.2 µs per mile.

Another point to keep in mind when calculating propagation delay is that optical fibers are not always physically placed over the shortest path between two geographic points, especially over transoceanic links. Due to installation convenience, circuits may be hundreds or even thousands of miles longer than theoretically necessary.

Nonetheless, the G.114 realtime communications network latency budget of 150 ms allows for nearly 24,000 km or 15,000 miles worth of propagation delay (which is approximately 60% of the earth's circumference); the theoretical worst-case scenario (exactly half of the earth's circumference) would require only 126 ms of latency. Therefore, this latency target is usually achievable for virtually any two locations (via a terrestrial path), given relatively direct transmission paths; however, in some scenarios meeting this latency target may simply not be possible, due to the distances involved and the relative directness of their respective transmission paths. In such scenarios, if the G.114 150 ms one-way latency target cannot be met due to the distances involved, administrators should be aware that both the ITU and Cisco Technical Marketing have shown that realtime communication quality doesn't begin to significantly degrade until one-way latency exceeds 200 ms, as is illustrated in the ITU G.114 graph of realtime speech quality vs. absolute delay, which is reproduced as Figure 3-2.

Figure 3-2 ITU G.114 Graph of Realtime Speech Quality versus Latency



Note The context so far has been for WAN circuits over terrestrial paths; for satellite circuits, the expected latency can be in the area of 250-900 ms. For example, signals being relayed via geostationary satellites will need to be sent to an altitude of 35,786 km (22,236 miles) above sea level (from the equator) out into space and then back to Earth again. There is nothing an administrator can do to decrease latency in such scenarios, as there is little they can do about increasing the speed of light or radio waves. All that can be done to address the effect of latency in these scenarios is to educate the user-base so that realistic performance expectations are set.

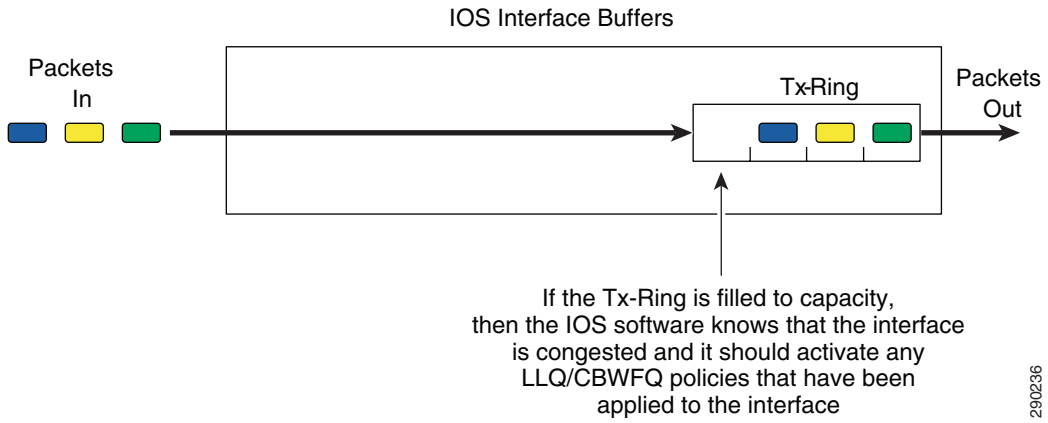
The final network latency component to be considered is queuing delay, which is variable (variable delay is also known as jitter). Queuing delay is a function of whether a network node is congested and if so, what scheduling policies have been applied to resolve congestion events. Realtime applications are often more sensitive to jitter than latency as a whole, as packets need to be received in de-jitter buffers prior to being played out. If a packet is not received within the window allowed for by the de-jitter buffer, it's as good as lost and can affect the overall voice or video call quality.

Given that the majority of factors contributing to network latency are fixed, careful attention has to be given to queuing delay, as this is the only latency factor that is directly under the network administrator's control-via their queuing policies. Therefore, a close examination of the IOS queuing system, including the Tx-Ring and LLQ/CBWFQ operation, will serve to assist network administrators optimize these critical policies.

Tx-Ring

The Tx-Ring is final IOS output buffer for a WAN interface (a relatively-small FIFO queue) that maximizes physical link bandwidth utilization by matching the outbound packet rate on the router with the physical interface rate. The Tx-Ring is illustrated in Figure 3-3.

Figure 3-3 IOS Tx-Ring Operation



The Tx-Ring also serves to indicate interface congestion to the IOS software. Prior to interface congestion, packets are sent on a FIFO basis to the interface via the Tx-Ring. However, when the Tx-Ring fills to its queue-limit, then it signals to the IOS software to engage any LLQ/CBWFQ policies that have been attached to the interface. Subsequent packets are then queued within IOS according to these LLQ/CBWFQ policies, dequeued into the Tx-Ring, and then sent out the interface in a FIFO manner.

The Tx-Ring can be configured on certain platforms with the **tx-ring-limit** interface configuration command. The value of the **tx-ring-limit** number can be from 1 to 32,767 packets. The default value of the Tx-Ring varies according to platform and link type and speed. The size of the Tx-ring can be ascertained with the **show controllers** *interface x/y* **| include tx** verification command, as shown in Example 3-1.

Example 3-1 Verifying the Tx-Ring Size—show controllers | include tx_limited

```
Router# show controllers Serial 1/0 | include tx_limited
tx_underrun_err=0, tx_soft_underrun_err=0, tx_limited=1(64)
Router#
```

In Example 3-1, the default value of the Tx-Ring is reported to be 64 packets on this serial interface.

During Cisco Technical Marketing design validation it was observed that the default Tx-Ring limit on some interfaces was shown to cause somewhat higher jitter values to some realtime application classes, particularly HD video-based realtime applications such as Cisco TelePresence traffic. The reason for this is the bursty nature of HD video traffic. For example, consider a fully-congested T3 WAN link (using a Cisco PA-T3+ port adaptor interface) with active LLQ/CBWFQ policies. The default Tx-Ring depth in this case is 64 packets, as shown in Example 3-1. Even if TelePresence traffic is prioritized via a LLQ, if there are no TelePresence packets to send, the FIFO Tx-Ring is filled with other traffic to a default depth of 64 packets. When a new TelePresence packet arrives, even if it gets priority treatment from the Layer 3 LLQ/CBWFQ queuing system, the packet are dequeued into the FIFO Tx-Ring when space is available. However, with the default settings, there can be as many as 63 packets in the Tx-Ring in front of that TelePresence packet. In such a worst-case scenario it could take as long as 17 ms to transmit these non-realtime packets out of this (45 Mbps) T3 interface. This 17 ms of instantaneous and variable delay (i.e., jitter) can affect the video quality for TelePresence to the point of being visually apparent to the end user. However, lowering the value of the Tx-Ring on this link will force in the IOS software engaging congestion management policies sooner and more often, resulting in lower overall jitter values for realtime applications, like TelePresence.

On the other hand, setting the value of the Tx-Ring too low may result in significantly higher CPU utilization rates, as the processor is being continually interrupted to engage queuing policies, even when congestion rates are just momentary bursts and not sustained rates. Thus when tuning the Tx-Ring, a trade-off setting is required such that jitter is minimized, but not at the expense of excessive CPU utilization rates.

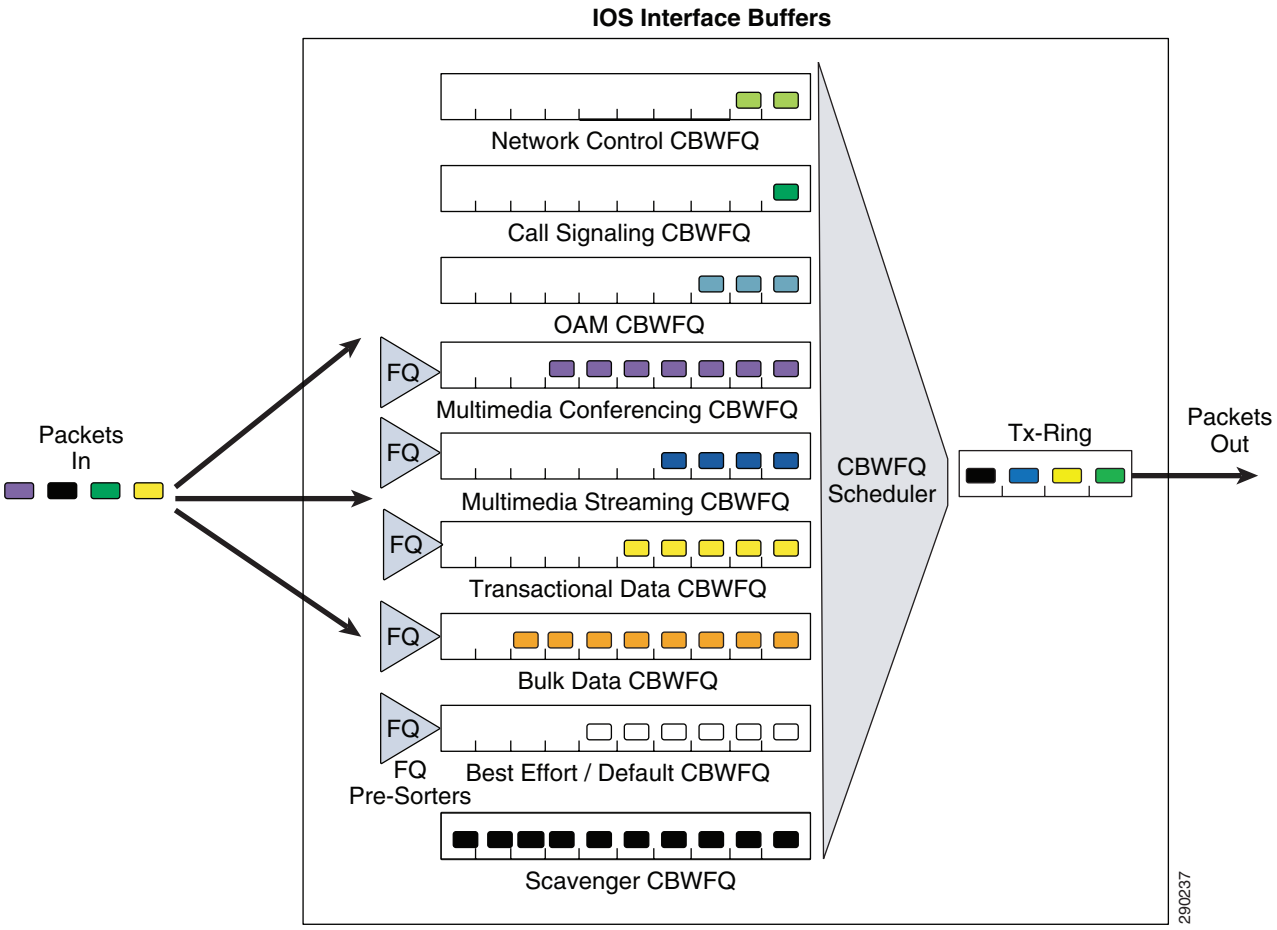
Therefore, in the link-specific design configurations in this chapter, explicit attention will be given to link types and speeds where the Tx-Ring is recommended to be tuned away from default values.

Class-Based Weighted-Fair Queuing

Class-Based Weighted-Fair Queuing (CBWFQ) is an IOS queuing algorithm that combines the ability to guarantee bandwidth with the ability to dynamically ensure fairness to other flows within a class of traffic.

The IOS software engages CBWFQ policies-provided these have been attached to an interface-only if the Tx-Ring for the interface is full, which occurs only in the event of congestion. Once congestion has thus been signaled to the software, it assigns each CBWFQ class is assigned its own queue. CBWFQ queues may also have a fair-queuing pre-sorter applied to them, such that multiple flows contending for a single queue are managed fairly. Additionally, each CBWFQ queue is serviced in a Weighted-Round-Robin (WRR) fashion based on the bandwidth assigned to each class. The CBWFQ scheduler then forwards packets to the Tx-Ring. The operation of CBWFQ is illustrated in Figure 3-4.

Figure 3-4 IOS CBWFQ Operation



Each CBWFQ class is guaranteed bandwidth via a bandwidth policy-map class configuration statement. CBWFQ derives the weight for packets belonging to a class from the bandwidth allocated to the class. CBWFQ then uses the weight to ensure that the queue for the class is serviced fairly, via WRR scheduling. The amount of bandwidth per-CBWFQ classes can be expressed in three different ways:

- **bandwidth** [*kbps*]—This is an absolute, unvarying value of guaranteed minimum bandwidth assigned to the queue, expressed in kilobits-per-second.
- **bandwidth percent** [*percent*]—This is a relative, unvarying value of guaranteed minimum bandwidth assigned to the queue, expressed as a percentage of the link's total bandwidth capacity (a fixed value); this syntax allows for greater modularity of CBWFQ policies, as these can be applied to different link-speeds and still be consistent and compatible; furthermore this syntax allows for link speeds to be changed (upgraded or downgraded) without having to reconfigure the queuing policies applied to them; therefore, this is the preferred CBWFQ bandwidth syntax used in this design chapter (however, this is purely a matter of administrative preference).
- **bandwidth remaining** [*percent*]—This is a relative, varying value of guaranteed minimum bandwidth assigned to the queue, expressed as a percentage of the link's bandwidth after the LLQ has been serviced (a dynamic, varying value); this syntax is generally preferred in service provider queuing scenarios.

An important point regarding bandwidth assigned to a given CBWFQ class is that the bandwidth allocated is *not a static bandwidth reservation*, but rather represents a *minimum bandwidth guarantee* to the class, provided there are packets offered to the class. If there are no packets offered to the class, then the scheduler services the next queue and can dynamically redistribute unused bandwidth allocations to other queues, as necessary.

Additionally, a fair-queuing pre-sorter may be applied to specific CBWFQ queues with the **fair-queue** policy-map class configuration command. It should be noted that this command enables a *flow-based* fair-queuing pre-sorter, and not a weighted fair-queuing pre-sorter, as the marketing name for this feature implies (and as such, the fair-queuing pre-sorter does not take into account the IP Precedence values of any packets offered to a given class). For example, if a CBWFQ class was assigned 1 Mbps of bandwidth and there were 4 competing traffic flows contending for this class, a fair-queuing pre-sorter would ensure that each flow receives (1 / (total-number-of-flows)) of bandwidth, or in this example (1/4 of 1 Mpbs) 250 kbps of bandwidth.

Note Prior to 12.4(20)T, a **fair-queue** pre-sorter could only be applied to class-default; however, since this software release, IOS includes the support of the Hierarchical Queuing Framework (HQF) which—among many other QoS feature enhancements—allows for a fair-queue pre-sorter to be applied to any CBWFQ class. HQF details are documented at: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/qos_frhqf_support_ps10591_TSD_Products_Configuration_Guide_Chapter.html.

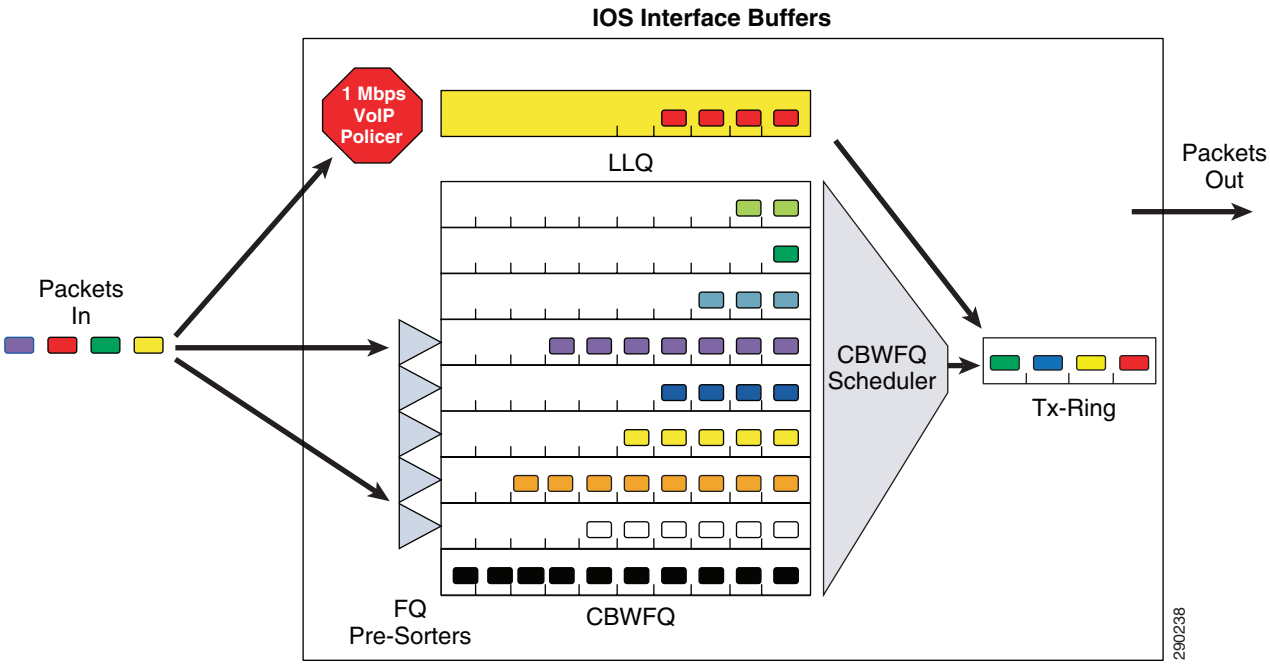
The depth of a CBWFQ is defined by its queue-limit, which varies according to link-speeds and platforms. This queue limit can be modified with the **queue-limit** policy-map class configuration command. The queue-limit is typically expressed as a number of packets, although on some platforms it may be expressed in bytes or in milliseconds. In some cases, such as provisioning (bursty) TelePresence traffic in a CBWFQ, it is recommended to increase the queue-limit from the default value, as will be discussed in more detail later in this design chapter.

Finally, older (pre-HQF / pre 12.4(20)T) versions of IOS software include a legacy feature that disallows LLQ/CBWFQ policies from being attached to an interface if these policies explicitly allocate more than 75% of the interface's bandwidth to non-default traffic classes. This was intended as a safety feature that would always allow for the default class, as well as control-traffic classes, to receive adequate bandwidth, as well as provision for Layer 2 bandwidth overhead. This feature can be overridden by applying the **max-reserved-bandwidth** interface command, which takes as a parameter the total percentage of interface bandwidth that can be explicitly provisioned (typically this value is set as 100). However, if this safety feature is overridden, then is highly-recommended that the default class be explicitly assigned no less than one-quarter of the link's bandwidth, as reviewed later in this design chapter.

Low-Latency Queuing

Low-Latency Queuing (LLQ) is essentially CBWFQ combined with a strict priority queue. Basic LLQ operation is illustrated in Figure 3-5.

Figure 3-5 IOS (Single) LLQ Operation



As shown in Figure 3-5, LLQ adds a strict-priority queue to the CBWFQ sub-system. The amount of bandwidth allocated to the LLQ is set by the **priority** policy-map class configuration command; bandwidth may be defined in absolute kbps values or as a percentage of the link's bandwidth, via the **percent** keyword (the latter is principally used throughout this design chapter, as this allows the LLQ policy to be more modular and adaptable; however, this is purely a matter of administrative preference).

An interesting facet of IOS LLQ is the inclusion of an implicit policer that admits packets to the strict-priority queue. This implicit policer limits the bandwidth that can be consumed by servicing the realtime queue and thus prevents bandwidth starvation of the non-realtime flows serviced by the CBWFQ scheduler. The policing rate for this implicit policer is always set to match the bandwidth allocation of the strict-priority queue; if more traffic is offered to the LLQ class than it has been provisioned to accommodate, then the excess traffic will be dropped by the policer. And like the LLQ/CBWFQ systems, the implicit policer is only active during the event of congestion (as signaled to the IOS software by means of a full Tx-Ring).

The functionality offered by the implicit LLQ policer is not limited to preventing bandwidth starvation of non-realtime traffic by realtime flows, but also can be leveraged to prevent multiple types of realtime flows from interfering with each other. For example, as shown in Figure 1-25 in Chapter 1, "Enterprise Medianet Quality of Service Design 4.0—Overview," there are up to three classes of traffic that may be provisioned with an Expedite Forwarding Per Hop Behavior (EF PHB), as outlined in RFC 4594: VoIP, Broadcast Video and Realtime-Interactive. However, if these three traffic classes were provisioned with a single **priority** statement then bursts from the Broadcast Video and/or Realtime-Interactive class could potentially interfere with (the better behaved) VoIP class. On the other hand, if each of these classes were provisioned with a multiple **priority** statements, as demonstrated in Example 3-2, then each of these classes would be metered by a dedicated implicit policer to ensure that adequate strict-priority queuing is guaranteed per EF class.

Example 3-2 Multiple LLQ Policy Example

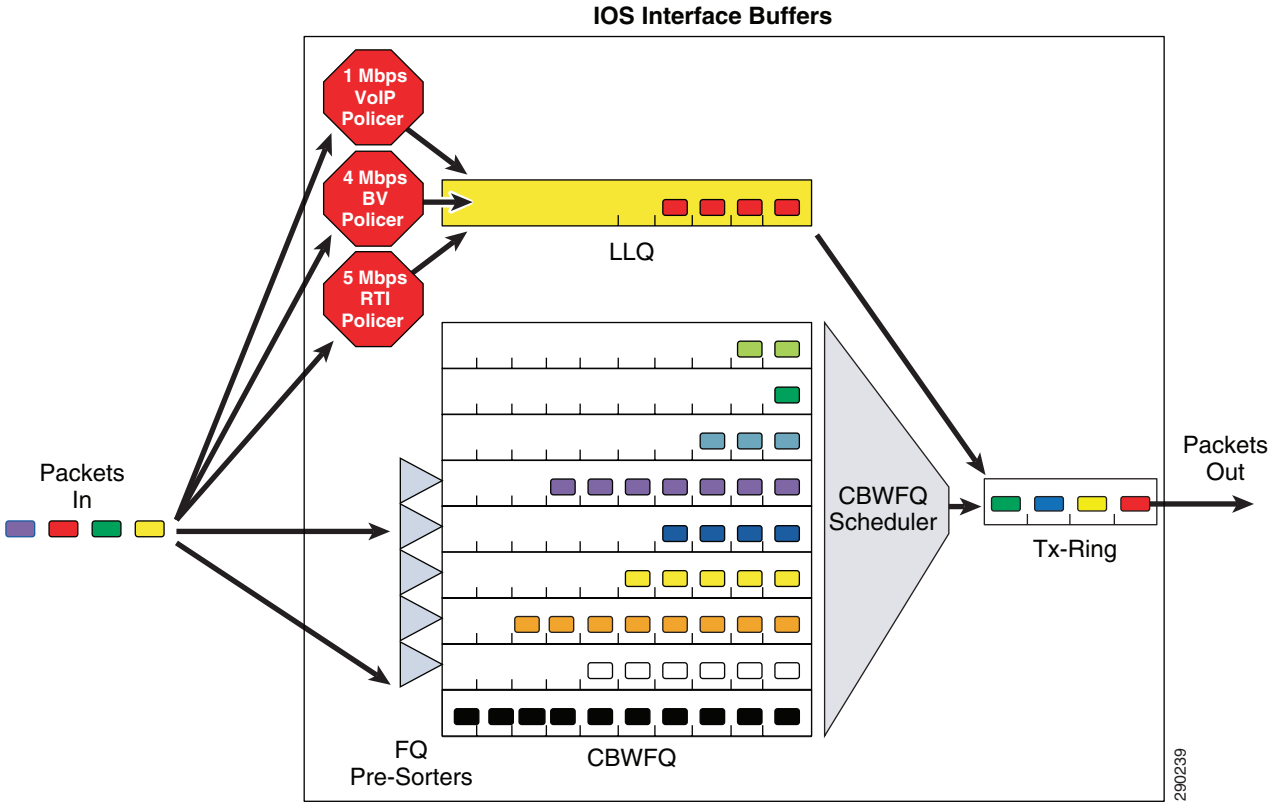
```
! This section defines the class-maps
class-map match-all VOIP
```

```
 match dscp ef
class-map match-all BROADCAST-VIDEO
 match dscp cs5
class-map match-all REALTIME-INTERACTIVE
 match dscp cs4


! This section defines the multi-LLQ policy-map
policy-map WAN-EDGE
 class VOIP
  priority 1000
 class BROADCAST-VIDEO
  priority 4000
 class REALTIME-INTERACTIVE
  priority 5000
...
```

The operation of the policy presented in Example 3-2 is illustrated in Figure 3-6.

Figure 3-6 IOS (Multiple) LLQ Operation



As shown in Figure 3-6, three separate implicit policers are enabled with this policy: one each for VoIP (at 1 Mbps), Broadcast Video (at 4 Mbps), and Realtime-Interactive (at 5 Mbps). However, it should be noted, that despite the configuration of multiple **priority** statements, there is still only a single strict-priority queue enabled within IOS. The total amount of bandwidth permitted for this queue is governed by the sum of the implicit policers, which in this case would be 10 Mbps (1 Mbps for VoIP + 4 Mbps for Broadcast Video + 5 Mbps for Realtime Interactive). Nonetheless, as reviewed later in more detail in this design chapter, it is recommended that the total amount of bandwidth allocated to priority queuing not exceed 33% of the link's capacity; in a multi-LLQ context, this would mean that the sum of all **priority** statements should be within one-third of the link's capacity.

Traffic vying for strict-priority queuing will be serviced on a first-come, first-serve basis, provided the packets are admitted by their respective policers. Thus, there is no operational advantage or disadvantage in the order of configuration of LLQ classes. In other words-referring to the previous example-the VoIP class is not serviced ahead of Broadcast Video or Realtime Interactive, but rather each class is serviced on a first-come, first-serve basis until their policer limits are reached.

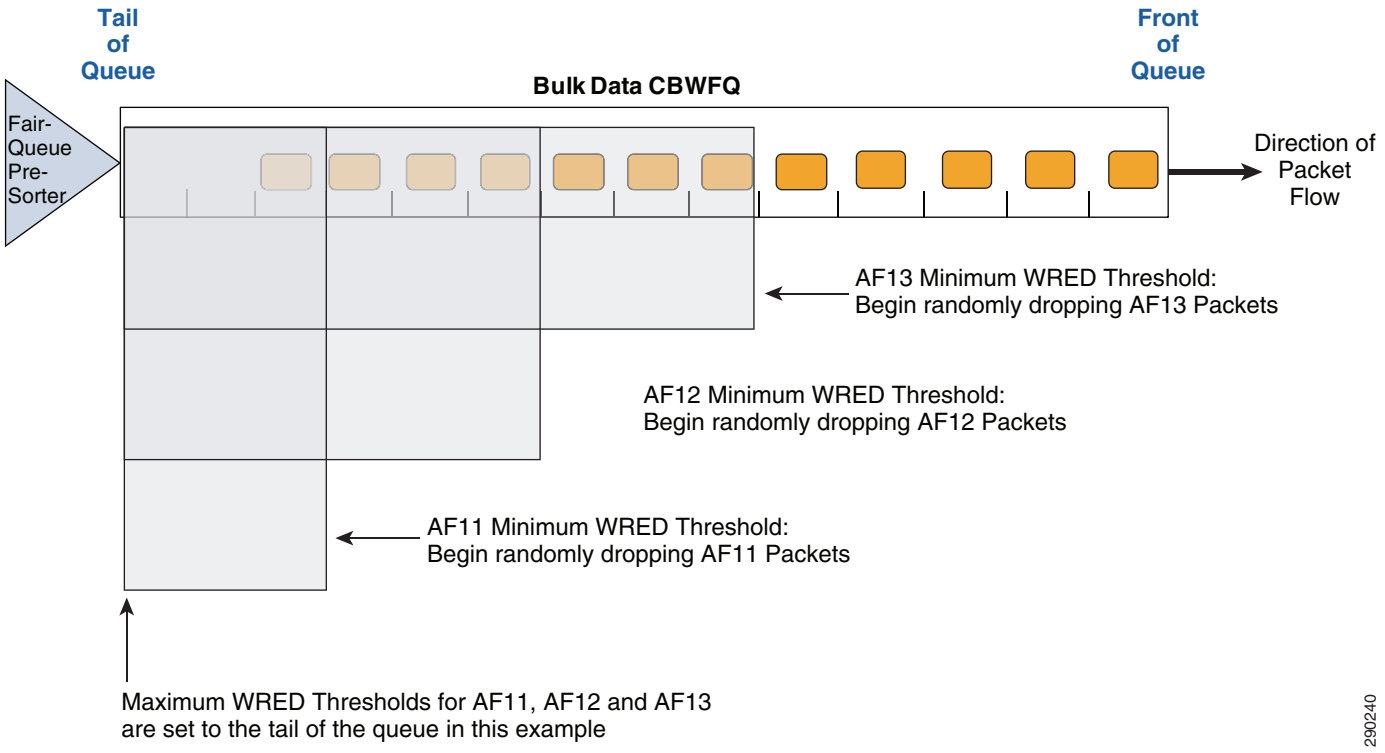
A final consideration with LLQ is the configuration of a burst parameter of the implicit LLQ policer, which is configured as an optional parameter of the **priority** policy-map class configuration command. Because traffic rates are not constant, especially for applications like video, packets may be presented to the LLQ implicit policers in sub-second bursts. For example, if 5 Mbps of traffic is provisioned for Realtime-Interactive flows, then-assuming a constant rate traffic flow-the policer would expect 625 Bytes per millisecond (50 Mbps/1000 milliseconds per second/8 bits per Byte). However, for a bursty Realtime-Interactive application-such as Cisco TelePresence-this constant sub-second traffic rate will not be the norm, but rather sub-second bursting will be more typical. Specifically, a single Cisco TelePresence camera can produce traffic that bursts as high as 64 Kbytes within a 33 ms video frame window. Therefore, to accommodate such applications, the burst parameter of the implicit LLQ policer may need to be adjusted, as will be detailed later in this design chapter. By default, the LLQ burst parameter computed as 200 milliseconds of traffic at the configured bandwidth rate and is used when the burst argument is not specified. The range of the burst is from 32 to 2000000 bytes.

Weighted-Random Early Detect

While congestion management mechanisms like LLQ/CBWFQ manage the front of the queue, congestion avoidance mechanisms like Weighted-Random Early Detect (WRED) manage the tail of the queue. Congestion avoidance mechanisms work best with TCP-based applications because selective dropping of packets causes the TCP windowing mechanisms to “throttle-back” and adjust the rate of flows to manageable rates.

The primary congestion avoidance mechanism in IOS is WRED, which randomly drops packets as queues fill to capacity. However, the randomness of this selection can be skewed by traffic weights. The weight can either be IP Precedence values, as is the case with default WRED which drops lower IPP values more aggressively (for example, IPP 1 would be statistically dropped more aggressively than IPP 6) or the weights can be AF Drop Precedence values, as is the case with DSCP-Based WRED which statistically drops higher AF Drop Precedence values more aggressively (for example, AF23 is dropped more aggressively than AF22, which in turn is dropped more aggressively than AF21). DSCP-based WRED is enabled with the **dscp** keyword in conjunction with the **random-detect** policy-map class configuration command. The operation of DSCP-based WRED is illustrated in Figure 3-7.

Figure 3-7 IOS DSCP-Based WRED Operation



As shown in Figure 3-7, packets marked with a given Drop Precedence (AFx3, AFx2 or AFx1) will only begin to be dropped when the queue fills beyond the minimum WRED threshold for the Drop Precedence value. Packets are always dropped randomly, but their probability of being dropped increases as the queue fills nearer the maximum WRED threshold for the Drop Precedence value. The maximum WRED thresholds are typically set at 100% (the tail of the queue), as shown; but these are configurable, and as such some advanced administrators may tune these WRED thresholds according to their needs, constraints and preferences.

WRED can also be used to set the RFC 3168 IP Explicit Congestion Notification (ECN) bits to indicate that congestion was experienced in transit. ECN functionality is enabled with the **ecn** keyword in conjunction with the **random-detect** policy-map class configuration command.

Link Types and Speeds

The link types and speeds that are supported on the WAN Aggregator platforms discussed in this design chapter (namely the Cisco 7200VXR G2, the Cisco Catalyst 6500 with SIP/SPA and the Cisco ASR family-at the time of writing) are summarized in Table 3-2.

Table 3-2 WAN Aggregation Link Types and Speeds

Media	Line Rates
Serial	T3 (45 Mbps)
ATM	DS3 (45 Mbps) to OC48/STM16 (2.5 Gbps)
POS	OC3/STM1 (155 Mbps) to OC192/STM64 (10 Gbps)
Ethernet	10 Mbps to 10 Gbps

It is important to note that while these platforms can support serial links as slow as DS0 (64 kbps) line rates, such rates are far too slow to support rich media applications. Therefore the minimum WAN aggregation line rate for medianet networks is recommended to be DS3/T3 (45 Mbps) in North America or E3 (34.368 Mbps) in Europe and elsewhere. However, lower speed links may be used at branch networks, as will be discussed in a subsequent design chapter. These minimum recommended line rates for medianet WAN networks render the use of (computationally-intensive) link-efficiency mechanisms, like IP RTP header compression (cRTP) and Link-Fragmentation and Interleaving (LFI), not only unnecessary, but wasteful.

Additionally, as previously mentioned, the scope of this design chapter is to cover WAN design fundamentals and designs specific to Layer 2 private WANs, with MPLS VPNs and Metro Ethernet VPNs being discussed in a following design chapter. Therefore only serial, ATM and POS links designs will be presented in this chapter, with Ethernet-based WAN designs being covered in the MPLS/Metro Ethernet design chapter.

PAK_Priority

PAK_priority is the internal Cisco IOS mechanism for protecting routing and control traffic, such as Interior Gateway Protocol (IGP) traffic (which includes RIP, EIGRP, OSPF and IS-IS). The PAK_priority feature is non-configurable and is not tunable in any way. The design implications of PAK_priority are summarized in the following list:

- PAK-priority traffic (Layer 2 and Layer 3 control traffic) is marked internally by the originating router as DSCP CS6 and is serviced by a dedicated, non-configurable set of system queues.
- On moderately congested WAN links control traffic is typically is protected adequately with the default PAK_priority treatment within the router.
- On heavily congested links, it is recommended explicitly provision a CBWFQ bandwidth class for routing/control traffic (identified by DSCP CS6), as this will indirectly allocate additional bandwidth for servicing the PAK_priority system queues.
- Although IS-IS traffic receives PAK_priority within the router, it cannot be marked to DSCP CS6 because IS-IS uses a CLNS protocol. (IS-IS does not use IP, so there are DSCP fields to mark.) This is important to keep in mind if explicit bandwidth provisioning is required for IS-IS traffic because it cannot be matched against CS6 like most other IGPs. However, NBAR can be used within a class map to match IS-IS traffic (for example, **match protocol clns_is**).
- Although BGP (both eBGPs and iBGPs) packets are marked by the originating router to CS6, they do not receive PAK_priority queuing within the routers. Therefore, it is recommended to provision a separate CBWFQ bandwidth class to protect BGP sessions, even on moderately congested links where the underlying IGPs are stable.
- On Catalyst 6500 switches running Cisco IOS Software on both the supervisors and MSFC, IGP packets marked internally with PAK_priority additionally are marked with DSCP CS6 and the Layer 2 CoS value of 6. This is because scheduling and congestion avoidance within (most) Cisco Catalyst 6500 switches and linecards is performed against Layer 2 CoS values.

Additionally, having an awareness of the PAK_priority feature helps in understanding some verification command output, such as output from the **show policy interface** command, which may show matches on a network control traffic class (as identified by DSCP CS6), but not show packets being queued for the CBWFQ assigned to that class (as these packets would instead be serviced by PAK_priority dedicated system queues).

Resource Reservations

So far, the QoS tools and designs discussed have all fallen under the Differentiated Services (DiffServ) architecture, where network devices perform classification, marking, policing, queuing and dropping services on a hop-by-hop basis; the DiffServ architecture operates in the data plane. An older QoS architecture is the Integrated Services (IntServ) architecture, where network devices make requests for static bandwidth reservations and maintain the state of all reserved flows while performing classification, marking and queuing services these flows; the IntServ architecture operates-and integrates-both the control plane and the data plane, and as such has been largely abandoned due to inherent scaling limitations.

Nonetheless, while the DiffServ Architecture is more scalable than the IntServ architecture, it lacks any ability to provide admission control. For example, consider the case of a (DiffServ) LLQ policy which has been provisioned to accommodate enough bandwidth to support 10 VoIP calls. If an 11th call is made and allowed onto the network, then since it-like the 10 previous calls before it-is correctly marked as EF, it will be offered to the LLQ. However, since the LLQ only has enough bandwidth to support 10 calls, the implicit policer will begin to drop packets from all calls-not just the last call attempted-resulting in reduced call quality for all users. DiffServ architectures require admission control decisions to be handled by a higher layer service.

Admission control mechanisms can fall into two broad categories:

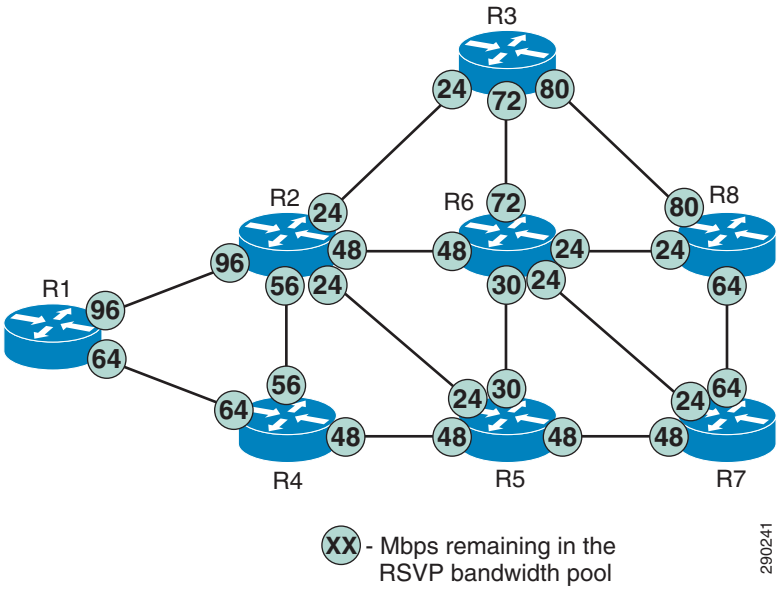
- Off-path—Examples include locations-based admission control and endpoint measurement-based admission control); these mechanisms can only support limited network topologies and have no network awareness.
- On-path—Primarily refers to Resource Reservation Protocol (RSVP)-based admission control; these mechanisms are can support a virtually any network topology and are network aware, and as such can dynamically adapt to network events.

Despite the advantages of (dynamic) on-path admission control mechanisms, most network administrators have preferred to deploy simpler (static) off-path admission control mechanisms to manage their VoIP deployments. However, the case for admission control mechanisms has become more urgent and complex with the advent of video-based media applications, as these applications have significantly higher (and burstier) bandwidth requirements on a per-flow basis. For example, consider the case of a Cisco TelePresence (CTS) deployment, where just one call from a CTS-3000 series endpoint to another can consume over 20 Mbps-in each direction. WAN bandwidth can be quickly consumed if intelligent admission control decisions are not made in realtime.

Before presenting design principles for RSVP-based on-path admission control, it would be beneficial to review the basic operation of RSVP, as it's a protocol that has been largely ignored for a number of years.

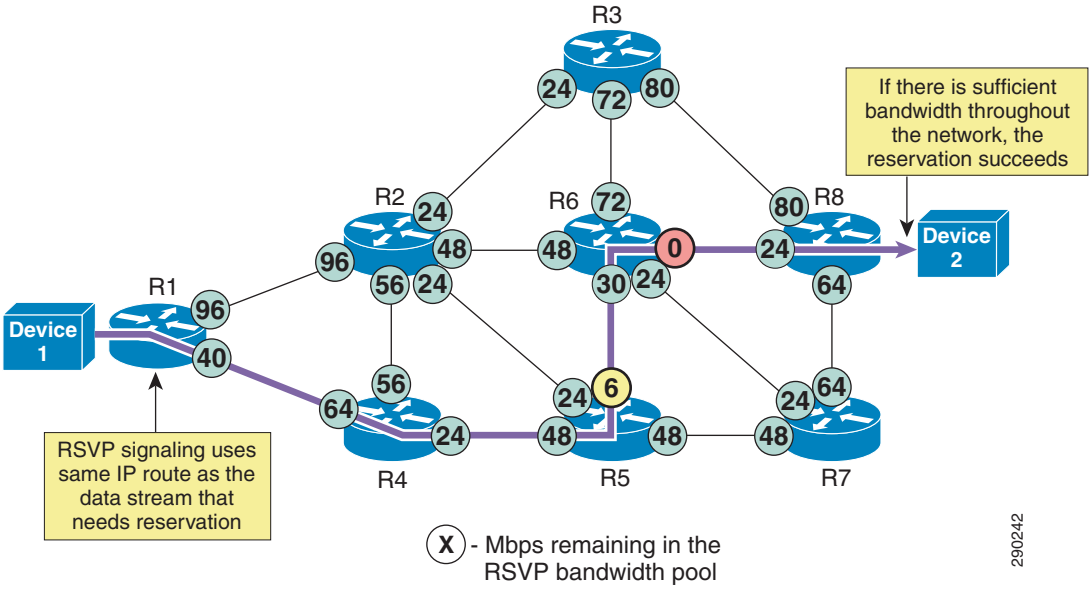
To better understand the basic principles of how RSVP performs bandwidth reservation in a network, consider the simple example depicted in Figure 3-8. This example does not analyze the exact message exchanges and protocol behaviors, but rather highlights the end results from a functionality perspective. Assume that RSVP is enabled on each router interface in the network shown in Figure 3-8 and that the numbers shown in the circles represent the amount of available RSVP bandwidth remaining on each interface.

Figure 3-8 Sample Network to Show RSVP Principles of Operation



Now consider an RSVP-enabled application that wants to reserve a certain amount of bandwidth for a media stream between two devices. This scenario is depicted in Figure 3-9, which shows a particular data stream that requires 24 Mbps of bandwidth from Device 1 to Device 2.

Figure 3-9 RSVP Functionality for a Successful Reservation



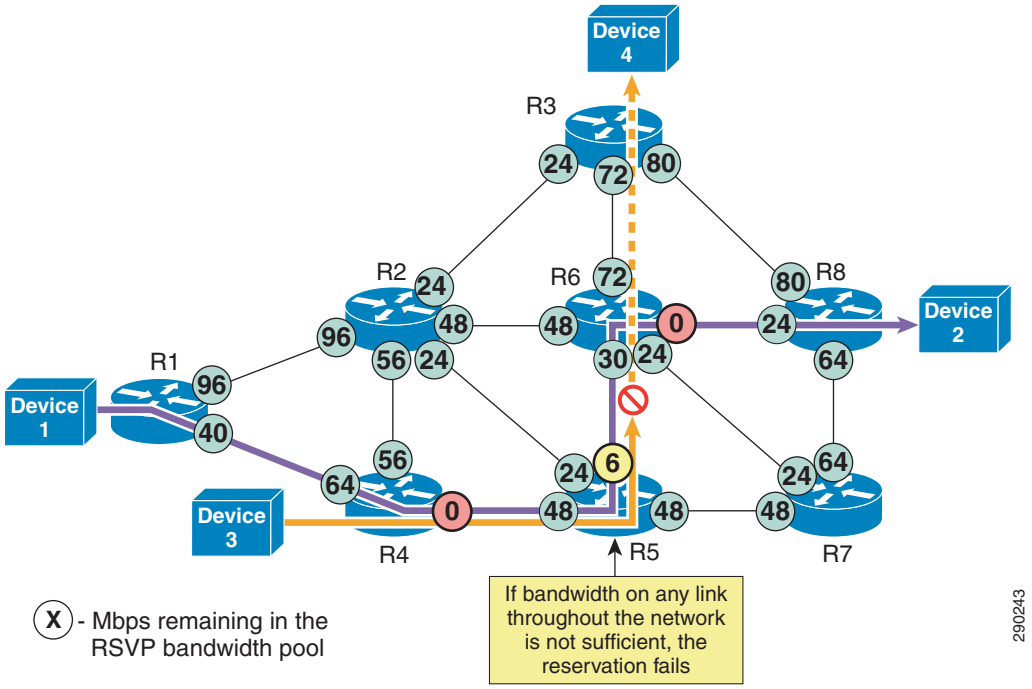
The following considerations apply to Figure 3-9:

- RSVP does not perform its own routing; instead it uses underlying routing protocols to determine where it should carry reservation requests. As routing protocols change paths to adapt to topology changes, RSVP adapts its reservations to the new paths wherever reservations are in place.

- The RSVP protocol attempts to establish an end-to-end reservation by checking for available bandwidth resources on all RSVP-enabled routers along the path from Device 1 to Device 2. As the RSVP messages progress through the network, the available RSVP bandwidth gets decremented by 24 Mbps on the outbound router interfaces, as shown in Figure 3-9.
- The available bandwidth on all outbound interfaces is sufficient to accept the new media stream, so the reservation succeeds and the application is notified.
- RSVP reservations are unidirectional (in this case, the reservation is established from Device 1 to Device 2, and not vice versa). In the presence of bidirectional applications such as voice and videoconferencing, two reservations must be established, one in each direction.
- RSVP provides transparent operation through router nodes that do not support RSVP. If there are any routers along the path that are not RSVP-enabled, they simply ignore the RSVP messages and pass them along like any other IP packet, and a reservation can still be established. However, in order to have an end-to-end QoS guarantee, you have to ensure that there is no possibility of bandwidth congestion on the links controlled by the non-RSVP routers.

After a reservation has been successfully established between Device 1 and Device 2, now assume that another application requests a 24 Mbps reservation between Device 3 and Device 4, as depicted in Figure 3-10.

Figure 3-10 RSVP Functionality for an Unsuccessful Reservation



The following considerations apply to Figure 3-10:

- The RSVP protocol attempts to establish an end-to-end reservation by checking for available bandwidth resources on all RSVP-enabled routers along the path from Device 3 to Device 4. As the RSVP messages progress through the network, the available RSVP bandwidth gets decremented by 24 Mbps on the outbound router interfaces, as shown in Figure 3-10.
- In this example, the available bandwidth on R5's outbound interface toward R6 is not sufficient to accept the new data stream, so the reservation fails and the application is notified. The available RSVP bandwidth on each outbound interface along the path is then restored to its previous value.

- The application can then decide what to do. It could abandon the data transfer or decide to send it anyway with no QoS guarantees, as best-effort traffic.

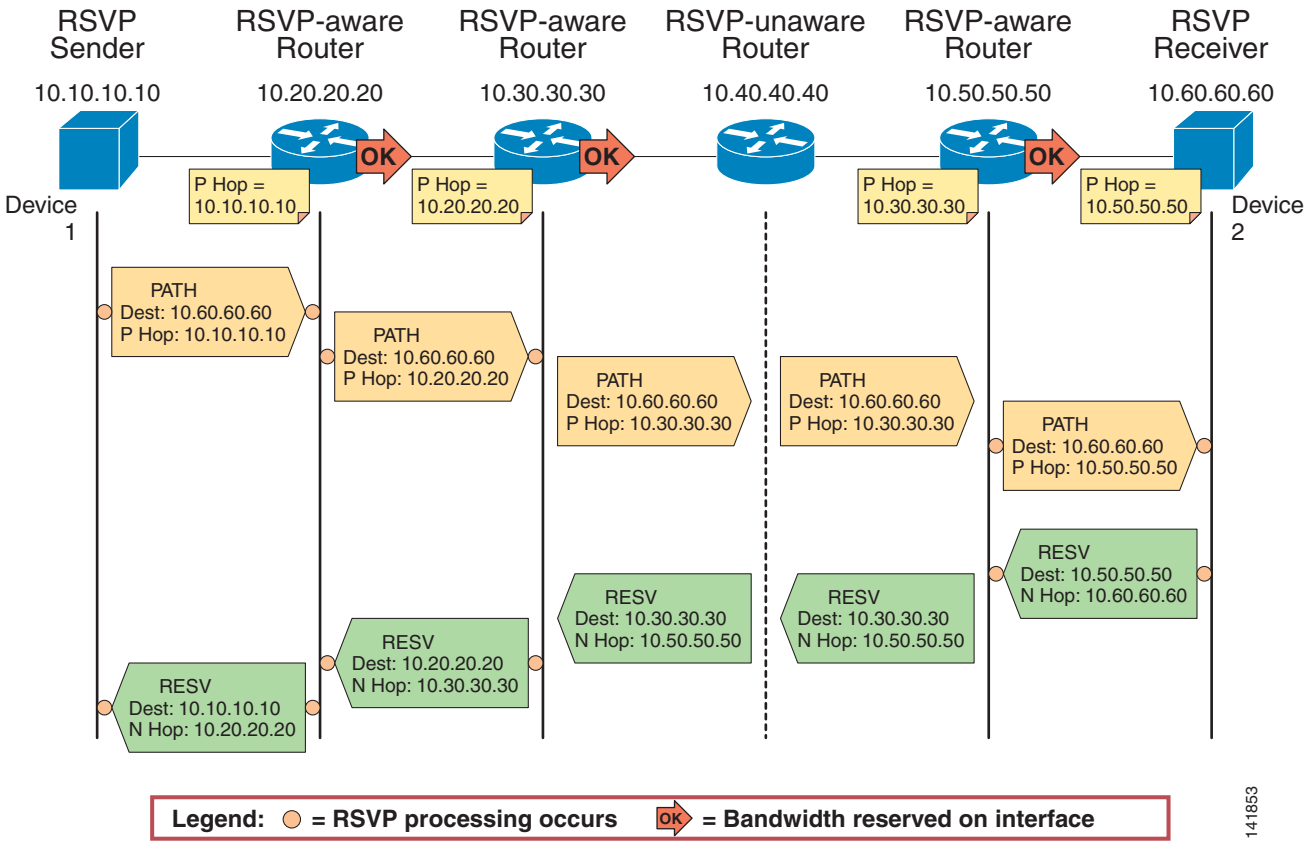
Thus having reviewed the high-level functionality of RSVP, it may be beneficial to examine the protocol in more detail to better understand how it interacts with the WAN infrastructure.

RSVP performs resource reservations for a given data flow across a network. RSVP reservations are unidirectional. Therefore, for a single audio call that contains two RTP streams, two RSVP reservations are generated, one for each RTP stream. The resource reservation is created by exchanging signaling messages between the source and destination devices for the data flow, and the messages are processed by intermediate routers along the path. The RSVP signaling messages are IP packets with the protocol number in the IP header set to 46 (this is not be confused with the IP DSCP value for VoIP packets, which are coincidentally also set to EF/46), and they are routed through the network according to the existing routing protocols.

Not all routers on the path are required to support RSVP because the protocol is designed to operate transparently across RSVP-unaware nodes. On each RSVP-enabled router, the RSVP process intercepts the signaling messages and interacts with the QoS manager for the router's outbound interface involved in the data flow in order to “reserve” bandwidth resources. When the available resources are not sufficient for the data flow anywhere along the path, the routers signal the failure back to the application that originated the reservation request.

The principles of RSVP signaling can be explained by using the example shown in Figure 3-11. In this diagram, an application wishes to reserve network resources for a data stream flowing from Device 1, whose IP address is 10.10.10.10, to Device 2, whose IP address is 10.60.60.60.

Figure 3-11 Example of RSVP Path and Resv Message Flow



The following steps describe the RSVP signaling process for as single data flow, as shown by the example in Figure 3-11:

1. The application residing on Device 1 originates an RSVP message called Path, which is sent to the same destination IP address as the data flow for which a reservation is requested (that is, 10.60.60.60) and is sent with the "router alert" option turned on in the IP header. The Path message contains, among other things, the following objects:
   - The "session" object, consisting of destination IP address, protocol number, and UDP/TCP port, which is used to identify the data flow in RSVP-enabled routers.
   - The "sender T-Spec" (traffic specification) object, which characterizes the data flow for which a reservation will be requested. The T-Spec basically defines the maximum IP bandwidth required for a call flow using a specific codec. The T-Spec is typically defined using values for the data flow's average bit rate, peak rate, and burst size.
   - The "P Hop" (or previous hop) object, which contains the IP address of the router interface that last processed the Path message. In this example, the P Hop is initially set to 10.10.10.10 by Device 1.
2. By means of the "router alert" option, the Path message is intercepted by the CPU of the RSVP-aware router identified as 10.20.20.20 in Figure 3-11, which sends it to the RSVP process. RSVP creates a path state for this data flow, storing the values of the session, sender Tspec, and P Hop objects contained in the Path message. Then it forwards the message downstream, after having replaced the P Hop value with the IP address of its outgoing interface (10.20.20.20 in this example).
3. Similarly, the Path message is intercepted by the CPU of the following RSVP-aware router, identified as 10.30.30.30 in Figure 3-11. After creating the path state and changing the P Hop value to 10.30.30.30, this router also forwards the message downstream.
4. The Path message now arrives at the RSVP-unaware router identified as 10.40.40.40 in Figure 3-11. Because RSVP is not enabled on this router, it just routes this message according to the existing routing protocols like any other IP packet, without any additional processing and without changing the content of any of the message objects.
5. Therefore, the Path message gets to the RSVP-aware router identified as 10.50.50.50, which processes the message, creates the corresponding path state, and forwards the message downstream. Notice that the P Hop recorded by this router still contains the IP address of the last RSVP-aware router along the network path, or 10.30.30.30 in this example.
6. The RSVP Receiver at Device 2 receives the Path message with a P Hop value of 10.50.50.50, and it can now initiate the actual reservation by originating a message called Resv. For this reason, RSVP is known as a receiver-initiated protocol. The Resv message carries the reservation request hop-by-hop from the receiver to the sender, along the reverse paths of the data flow for the session. At each hop, the IP destination address of the Resv message is the IP address of the previous-hop node, obtained from the path state. Hence, in this case Device 2 sends the Resv message with a destination IP address of 10.50.50.50. The Resv message contains, among other things, the following objects:
   - The "session" object, which is used to identify the data flow.
   - The "N Hop" (or next hop) object, which contains the IP address of the node that generated the message. In this example, the N Hop is initially set to 10.60.60.60 by Device 2.
7. When RSVP-aware router 10.50.50.50 receives the Resv message for this data flow, it matches it against the path state information using the received session object, and it verifies if the reservation request can be accepted based on the following criteria:
   - Policy control—Is this user and/or application allowed to make this reservation request?

  - Admission control—Are there enough bandwidth resources available on the relevant outgoing interface to accommodate this reservation request?

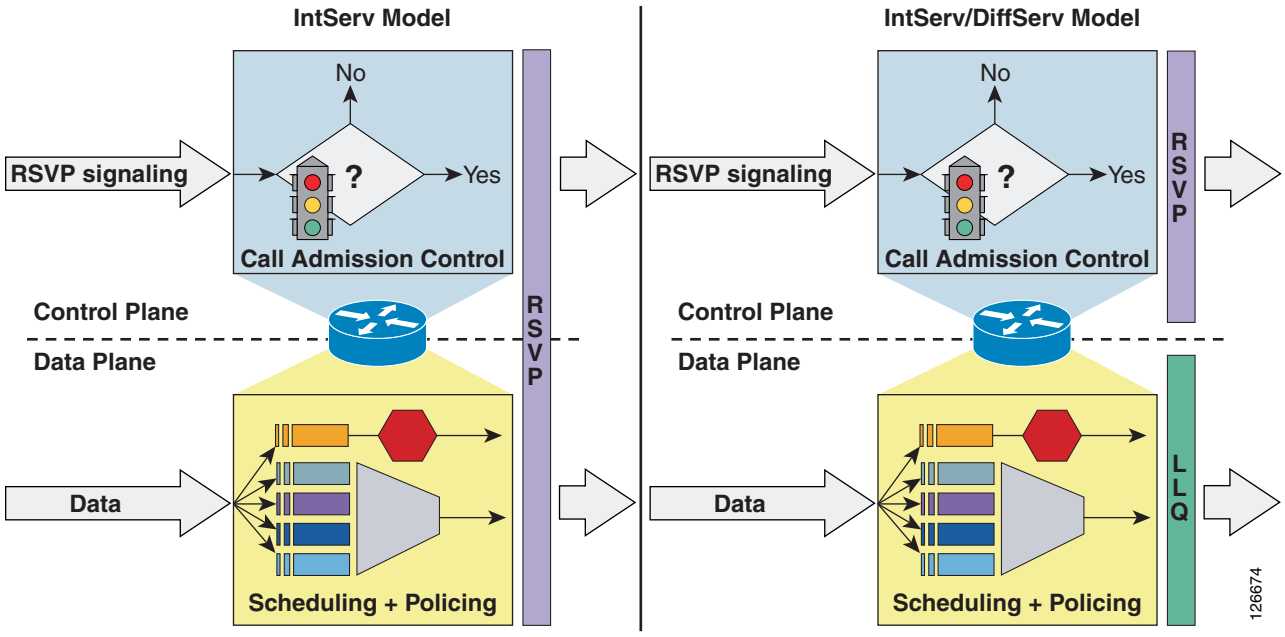
8. In this case, we assume that both policy and admission control are successful on 10.50.50.50, which means that the bandwidth provided by the Tspec in the path state for this session is reserved on the outgoing interface (in the same direction as the data flow, that is from Device 1 to Device 2), and a corresponding "reservation state" is created. Now router 10.50.50.50 can send a Resv message upstream by sending it as a unicast IP packet to the destination IP address stored in the P Hop for this session, which was 10.30.30.30. The N Hop object is also updated with the value of 10.50.50.50.
9. The Resv message now transits through the RSVP-unaware router identified as 10.40.40.40, which will route it toward its destination of 10.30.30.30 like any other IP packet. This mechanism allows RSVP signaling to work across a heterogeneous network where some nodes are not RSVP-enabled.
10. The RSVP-aware router identified as 10.30.30.30 receives the Resv message and processes it according to the mechanisms described in steps 7 and 8. Assuming policy and admission control are successful also at this hop, the bandwidth is reserved on the outgoing interface and a Resv message is sent to the previous hop, or 10.20.20.20 in this example.
11. After a similar process within the router identified as 10.20.20.20, the Resv finally reaches the RSVP sender, Device 1. This indicates to the requesting application that an end-to-end reservation has been established and that bandwidth has been set aside for this data flow in all RSVP-enabled routers across the network. This example shows how the two main RSVP signaling messages, Path and Resv, travel across the network to establish reservations. Several other messages are defined in the RSVP standard to address error situations, reservation failures, and release of resources. In particular, the ResvErr message is used to signal failure to reserve the requested resources due to either policy control or admission control somewhere along the network. If, for example, admission control had failed at node 10.50.50.50 in Figure 3-11, this node would have sent a ResvErr message back to Device 2, specifying the cause of the failure, and the application would have been notified.

Another important aspect of the RSVP protocol is that it adopts a soft-state approach, which means that for each session both the path state and the reservation state along the network need to be refreshed periodically by the application by sending identical Path and Resv messages. If a router does not receive refresh messages for a given session for a certain period of time, it deletes the corresponding state and releases the resources reserved. This allows RSVP to react dynamically to network topology changes or routing changes due to link failures. The reservations simply start flowing along the new routes based on the routing protocol decisions, and the reservations along the old routes time-out and are eventually deleted.

RSVP can be deployed in two operational models, as shown in Figure 3-12:

- IntServ Model—This model integrates the control plane and the data plane, with RSVP handling admission control, classification, marking, policing and scheduling operations. This is the legacy RSVP operational model, and as previously mentioned, has been largely abandoned due to inherent scalability limitations.
- IntServ/DiffServ Model—This model separates control plane operations from data plane operations. RSVP operation is limited to admission control only; with DiffServ mechanisms handling classification, marking, policing and scheduling operations. As such, the IntServ/DiffServ model is highly scalable and flexible. The designs presented in this chapter are based on the IntServ/DiffServ model.

Figure 3-12 RSVP Operational Models: IntServ and IntServ/DiffServ Models



As shown in Figure 3-12, under the IntServ/DiffServ model, RSVP is only utilized to perform admission control; all other QoS functions-including classification, marking, policing, shaping, queuing and dropping-are handled by DiffServ policies. This combination allows for efficient scaling of policies along with network aware admission control. The IntServ/DiffServ model was introduced in IOS 12.2(2)T as the RSVP Scalability Enhancements feature.

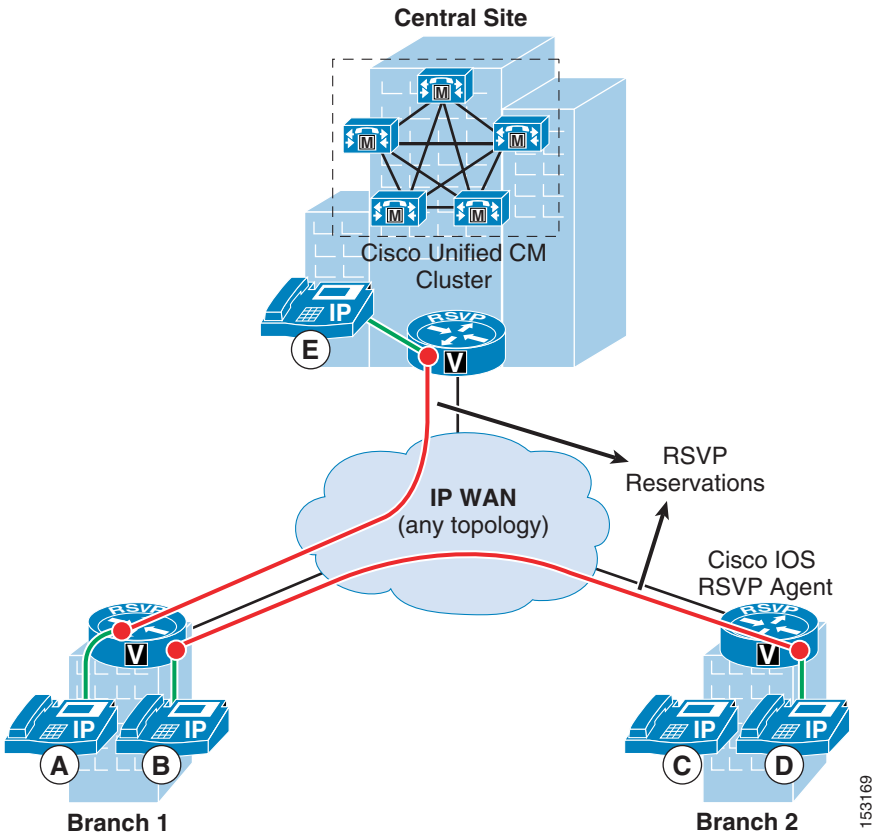
Before RSVP can be effectively enabled on a per-interface basis, the IOS RSVP Agent functionality must be enabled on a router. The Cisco RSVP Agent is a Cisco IOS feature that enables Cisco Unified CallManager (CUCM) to perform RSVP-based call admission control. The Cisco RSVP Agent feature was introduced in Cisco IOS Release 12.4(6)T.

The Cisco RSVP Agent registers with Unified CM as either a media termination point (MTP) or a transcoder device with RSVP support. When an endpoint device makes a call in need of a bandwidth reservation, CUCM invokes a Cisco RSVP Agent to act as a proxy for the endpoint to make the bandwidth reservation.

Figure 3-13 illustrates the high-level operation of the IOS RSVP Agent. Specifically, Figure 3-13 shows a typical Cisco RSVP Agent deployment within a CUCM cluster, which includes three locations: Central Site, Branch 1, and Branch 2. The IP WAN connecting the three locations can be of any topology type and is not restricted to the hub-and-spoke topology. For any call between two locations that requires an RSVP reservation in the media path, a pair of Cisco RSVP Agents is invoked dynamically by CUCM. The Cisco RSVP Agent acts as a proxy to make an RSVP reservation for the IP Phone in the same location with the Cisco RSVP Agent. For example, when phone A in Branch 1 calls phone E in the Central Site, an RSVP reservation (illustrated as the red line in Figure 3-13) is established between Cisco RSVP Agents in the Branch 1 and Central Site locations.

There are three call legs for the media streams of this call. The first call leg is between phone A and the Branch 1 Cisco RSVP Agent, the second call leg is between the Branch 1 and Central Site Cisco RSVP Agents, and the third call leg is between the Central Site Cisco RSVP Agent and phone E. By the same token, when phone B in Branch 1 calls phone D in Branch 2, the RSVP reservation is established between the Branch 1 and Branch 2 Cisco RSVP Agents. Note that the media streams of a call between two branch locations are not sent through the Central Site in this case, which is different from a call made over the traditional hub-and-spoke topology using call admission control based on static locations.

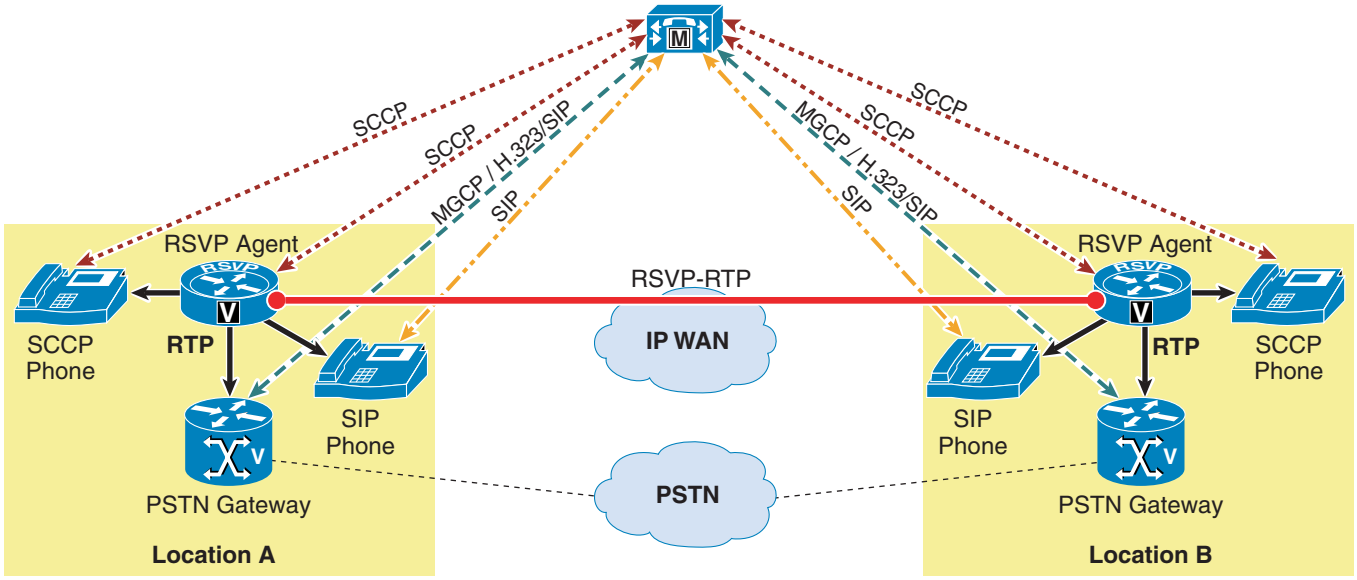
Figure 3-13 Cisco IOS RSVP Agent Operation



For any calls across the WAN, CUCM directs the endpoint devices to send the media streams to their local Cisco RSVP Agent, which originates another call leg synchronized with an RSVP reservation to the Cisco RSVP Agent at the remote location. Figure 3-14 illustrates the following signaling protocols used in CUCM and IOS RSVP Agent deployments:

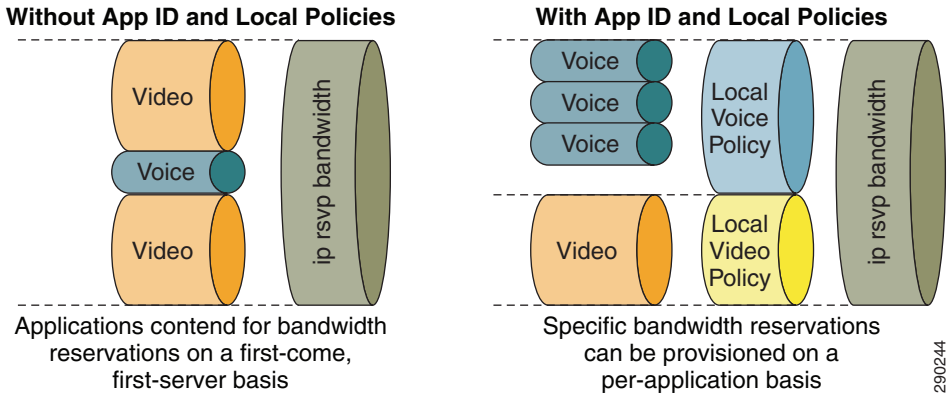
- Cisco RSVP Agents register to Unified CM via Skinny Client Control Protocol (SCCP).
- IP phones register with Unified CM via SCCP or Session Initiation Protocol (SIP).
- PSTN gateways register with Unified CM via Media Gateway Control Protocol (MGCP), SIP, or H.323 protocol.

Figure 3-14 Protocol Flows for Locations with RSVP



Furthermore, RSVP admission control policies can be even more granular by integrating Application Identification (App-ID) within the reservation requests (as based on RFC 2872). With such functionality, separate local (per-interface) RSVP policies can be configured to handle bandwidth requests on a per-application basis, rather than at an aggregate level alone. RSVP reservations with and without App-ID functionality are illustrated in Figure 3-15.

Figure 3-15 IOS RSVP App-ID Functionality



Without App-ID reservations, an administrator could allocate, for example, 5 Mbps total for voice and/or video calls, but these calls would get access to the reservable bandwidth on a first-come, first-served basis. On the other hand, with App-ID reservations, the administrator could specify that 1 Mbps of traffic is to be reserved for voice calls (only) and 4 Mbps for video calls (only).

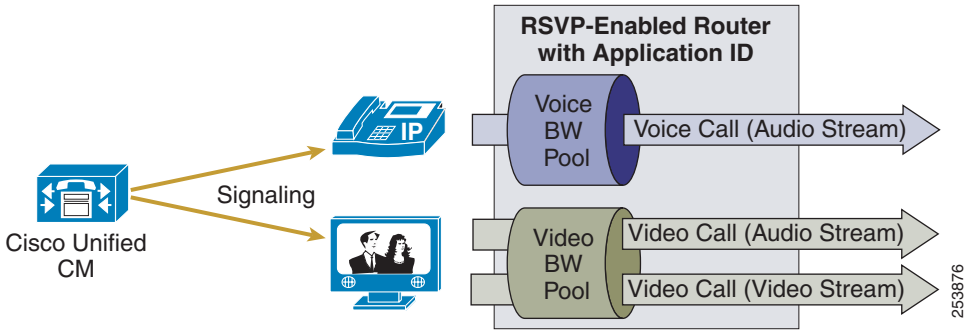
RSVP Application ID was introduced in IOS 12.4(6)T. The RSVP Application ID strings can be configured via two service parameters in the Cisco Unified CallManager (CUCM) cluster-wide RSVP parameter configuration, one each for voice and video:

- o RSVP Audio Application ID (Default = AudioStream)—When a voice call is made between locations with an RSVP policy, the resulting reservations for the audio stream will be tagged with the RSVP Audio Application ID.

- RSVP Video Application ID (Default = VideoStream)—When a video call is made between locations with an RSVP policy, the resulting reservations for the audio and video streams will both be tagged with the RSVP Video Application ID. A video call has both audio and video associated to the Video Application ID.

Figure 3-16 shows how CUCM tags voice and video calls with an Application ID in RSVP.

Figure 3-16 CUCM and RSVP Application ID Functionality



A final word on RSVP: It should be noted that expanding RSVP functionality and continuing to make it more adaptable and efficient is a major area of research and development within Cisco's medianet initiative. As such, network QoS design recommendations for medianet will have an ever greater RSVP component for admission and resource control.

Medianet WAN Interface Roles

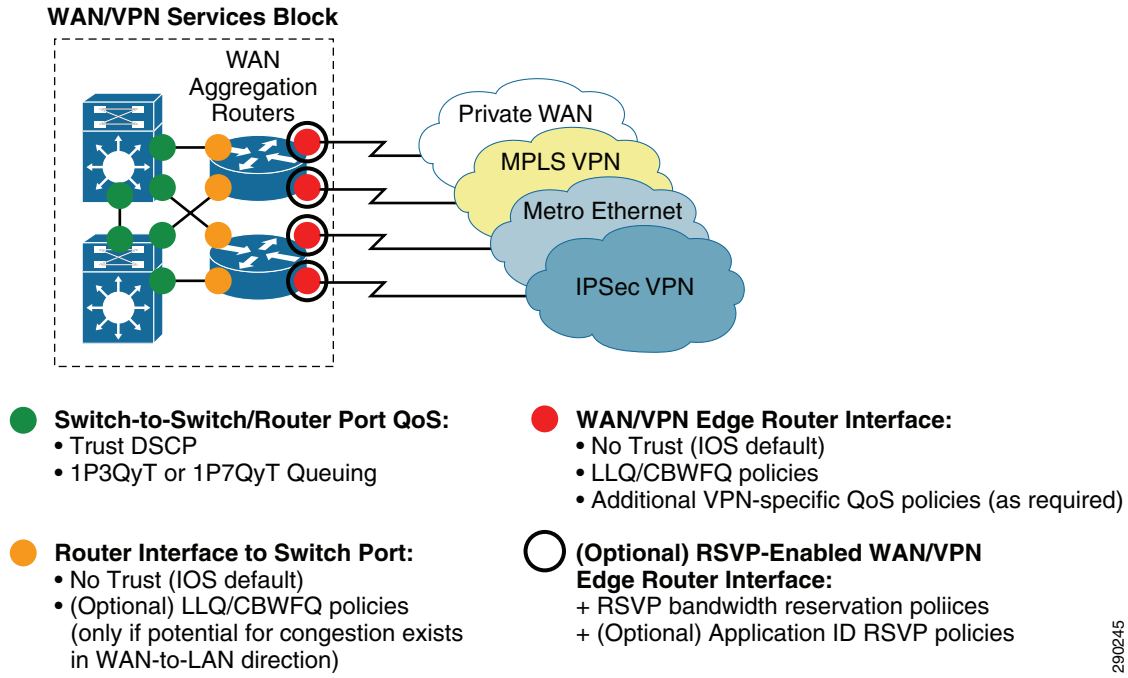
The policy elements discussed thus far can be grouped into roles that various router interfaces and switch ports serve within the medianet WAN architecture, such as:

- (Gigabit-Ethernet) Switch ports connecting to switch ports or router interfaces:
  - Static trust policies **should** be configured on these ports, preferably DSCP-trust for maximum classification and marking granularity.
  - Optional ingress marking or policing policies (such as data plane policing policies) **may** be configured on these ports.
  - Egress queuing policies that support (at a minimum) 1P3QyT queuing **should** be configured on these ports, preferably with DSCP-to-queue mapping. However, switch platforms/linecards that support 1P7QyT queuing are preferred at the distribution and core layers for increased queuing granularity at these aggregation layers.
  - Distribution downlinks (to the access layer) may be configured with microflow policing or User-Based Rate Limiting (UBRL) to provide a potential second line of policing defense for the medianet campus network.
- Router (Gigabit-Ethernet) LAN interfaces connecting to switch ports:
  - No explicit trust policies need be applied, as trust-functionality is effectively enabled by default within Cisco IOS.
  - Egress LLQ/CBWFQ policies **may** be applied on these interfaces; however, usually traffic from the WAN toward the LAN is insufficient to create a congestion/queuing scenario, and as such enabling queuing policies in this direction is optional. If egress queuing policies are applied, these should be consistent with WAN-edge queuing policies.

- Router WAN interfaces:
  - No explicit trust policies need be applied, as trust-functionality is effectively enabled by default within Cisco IOS.
  - Egress LLQ/CBWFQ policies **should** be applied on these interfaces, as these LAN/WAN edges will most often represent a speed-mismatch and thus a congestion/queuing scenario.
  - RSVP policies **may** be enabled on these interfaces to provide network-aware dynamic admission control functionality. RSVP policies **may** include Application ID RSVP policies.

Figure 3-17 shows these router interface and switch port QoS roles within a medianet WAN/VPN architecture.

Figure 3-17 Medianet WAN/VPN Router Interface and Switch Port QoS Roles



AutoQoS

Automatic QoS (AutoQoS) is essentially an intelligent macro that enables an administrator to enter one or two simple AutoQoS commands to enable all the appropriate features for the recommended QoS settings for an application on a specific interface. In Cisco IOS, there are (at the time of writing) two AutoQoS features available:

- AutoQoS-VoIP—Automatically provisions QoS policies and link-specific parameters for optimal deployment of IP Telephony QoS over router WAN links.
- AutoQoS-Enterprise—Automatically discovers traffic classes and can recommend and provision QoS policies and link-specific parameters for optimal deployment of multiple application classes over router WAN links.

Each of these AutoQoS features will be discussed in additional detail in the following sections.

AutoQoS-VoIP

The first IOS release of AutoQoS (beginning with Cisco IOS 12.2[15]T) was the AutoQoS-VoIP feature, which provides best-practice QoS configurations for IP Telephony on IOS routers. However, it is important to recognize that-as the feature-name implies-AutoQoS-VoIP is intended to support QoS for IP Telephony deployments only and it does not provision QoS for other medianet application classes.

Note The AutoQoS-VoIP feature is also available on Cisco Catalyst switches, as discussed in Chapter 2, "Medianet Campus QoS Design 4.0." Thus, with a single Auto-QoS command, complementary policies for IP Telephony can be deployed over both the LAN (on Cisco Catalyst switches) and the WAN (on Cisco IOS routers).

In Cisco IOS, AutoQoS-VoIP is supported on Frame Relay, ATM, HDLC, Point-to-Point Protocol (PPP), and Frame Relay-to-ATM links. AutoQoS-VoIP performs the following QoS functions automatically:

- Classifies and marks VoIP bearer traffic (to EF)
- Classifies and marks signaling protocol traffic (to CS3), including:
  - H323
  - H.225 (unicast only)
  - Session Initiation Protocol (SIP)
  - Skinny Call Control Protocol (SCCP)
  - Media Gateway Control Protocol (MGCP
- Applies scheduling policies, including:
  - Low-latency queuing (LLQ) for voice
  - Class-based weighted fair queuing (CBWFQ) for call signaling
  - Fair queuing (FQ) for all other traffic
- Enables Frame Relay traffic shaping with optimal parameters, if required
- Enables Link Fragmentation and Interleaving (either Multilink PPP (MLPPP) Link Fragmentation and Interleaving (LFI) or Frame Relay Fragmentation (FRF.12) on links with speeds less than or equal to 768 kbps, if required
- Enables IP RTP header compression (cRTP), if required
- Provides RMON alerts of VoIP packets that are dropped

The AutoQoS-VoIP feature is enabled in IOS routers by entering the **auto qos voip** interface configuration command, the AutoQoS-VoIP macro automatically configures the recommended VoIP QoS configurations (complete with all the calculated parameters and settings, as described above) for the interface on which the AutoQoS-VoIP feature is applied. Additionally, the AutoQoS-VoIP feature can take two optional parameters:

- **trust**—This parameter indicates that the DSCP markings of a packet are to be trusted for classification of the voice and signaling traffic. If the optional **trust** keyword is not specified, the voice traffic is classified using Network-Based Application Recognition (NBAR), and the packets are marked with the appropriate DSCP value.
- **ft-atm**—This parameter enables the AutoQoS-VoIP feature for Frame-Relay-to-ATM links. This option is available on the Frame Relay Data-Link Connection Identifiers (DLCIs) for Frame-Relay-to-ATM interworking only.

For additional details on AutoQoS-VoIP, refer to the feature documentation at: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/autoqos_voip_ps10591_TSD_Products_Configuration_Guide_Chapter.html.

AutoQoS-Enterprise

The second IOS release of AutoQoS (beginning with Cisco IOS 12.3[7]T) was the AutoQoS-Enterprise feature. AutoQoS-Enterprise performs all the QoS functionality of AutoQoS-VoIP, as well as it detects and provisions up to 10 application traffic classes The AutoQoS Enterprise feature consists of two configuration phases, completed in the following order:

- Auto-discovery (data collection)—This phase uses protocol discovery based on Network-Based Application Recognition (NBAR) to detect the applications on the network and perform statistical analysis on the network traffic.
- AutoQoS-Enterprise template generation and installation—This phase generates templates from the data collected during the auto-discovery phase and installs the templates on the interface. These templates then are used as the basis for creating the class maps and policy maps for the network. After the class maps and policy maps are created, they are installed on the relevant interface.

The AutoQoS for the Enterprise feature defines up to 10 application classes. Table 3-3 lists these AutoQoS-Enterprise application classes, along with the type of traffic defined for each class, and the DSCP value that AutoQoS-Enterprise assigns the class.

Table 3-3 Class Definitions for the AutoQoS-Enterprise Feature

AutoQoS-Enterprise Class Name	Traffic Type	DSCP Value
IP Routing	Network control traffic, such as routing protocols	CS6
Interactive Voice	Inactive voice-bearer traffic	EF
Interactive Video	Interactive video data traffic	AF41
Streaming Video	Streaming media traffic	CS4
Telephony Signaling	Telephony signaling and control traffic	CS3
Transactional/Interactive	Database applications transactional in nature	AF21
Network Management	Network management traffic	CS2
Bulk Data	Bulk data transfers; web traffic; general data service	AF11
Scavenger	Casual entertainment; rogue traffic; traffic in this category is given less-than-best-effort treatment	CS1
Best Effort	Default class; all non-critical traffic; HTTP; all miscellaneous traffic	0

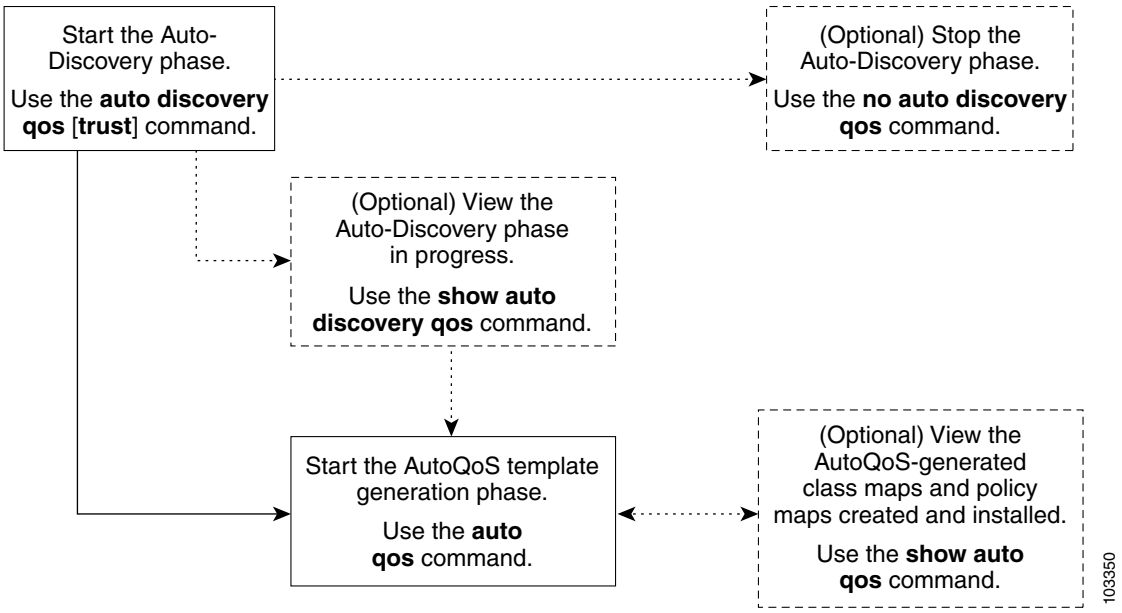
Note The individual applications per application class, along with corresponding NBAR Protocol Description Language Modules (PDLMs), are listed at: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/autoqos_enterprise_ps10591_TSD_Products_Configuration_Guide_Chapter.html#wp1158725.

Like AutoQoS-VoIP, the AutoQoS-Enterprise feature is supported on Frame Relay, ATM, HDLC, Point-to-Point Protocol (PPP), and Frame Relay-to-ATM links. The steps to deploying AutoQoS-Enterprise are as follows:

- First, start the Auto-Discovery (data collection) phase by using the **auto discovery qos** interface configuration command. Note the following points about the Auto-Discovery phase:
  - If you want to stop the Auto-Discovery phase, use the **no auto discovery qos** command. This command stops data collection and removes any data collection reports that have been generated.
  - If you want to view the Auto-Discovery phase in progress, use the **show auto discovery qos** command. This command displays the results of the data collected during the Auto-Discovery phase.
- Second, start the AutoQoS template generation phase by using the **auto qos** interface configuration command. This phase generates templates from the data collected during the Auto-Discovery phase. It then uses those templates as the basis for creating and installing the class maps and policy maps for your network.

These top-level processes for enabling AutoQoS-Enterprise are illustrated in the flowchart in Figure 3-18.

Figure 3-18 AutoQoS-Enterprise Top-Level Processes and Configuration



For additional details on AutoQoS-Enterprise, refer to the feature documentation at: http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/autoqos_enterprise_ps10591_TSD_Products_Configuration_Guide_Chapter.html.

AutoQoS Considerations

Some may naturally ask, why should I read this lengthy and complex QoS design document when I have AutoQoS?

Well, to answer this question, it is important to recognize the intent and capabilities of these AutoQoS features, as well as their limitations.

For instance, AutoQoS-VoIP is an excellent tool for customers who want to enable QoS for IP Telephony **only**, and who have only basic QoS needs or do not have the time or desire to do more with QoS. However, if business requirements necessitate more application classes that need to be provisioned with

QoS polices over the WAN, then the administrator needs to either do extensive customization to AutoQoS-VoIP policy templates or perhaps consider AutoQoS-Enterprise or even consider implementing QoS policies manually.

On the other hand, while the AutoQoS-Enterprise feature can accommodate additional traffic classes, it should be noted that there are three main discrepancies between AutoQoS-Enterprise traffic classes and Cisco Medianet traffic classes. Additionally, it is important to keep in mind that AutoQoS-Enterprise bases its configuration recommendation on statistical sampling of traffic; therefore, these samples may vary from link-to-link, as well as by time of day/month/year. Furthermore, sampling is a reactive approach to QoS policy design, and not a proactive one; after all, just because there is currently a set amount of traffic for a given class across the link doesn't necessarily mean that this is what is intended by the business requirements.

It is also important to remember how AutoQoS features have developed: AutoQoS features are the result of Cisco QoS feature development coupled with Cisco Validated Design guides (CVDs): AutoQoS-VoIP is the product of the first IP Telephony QoS design guide (published in 1999) and the AutoQoS-VoIP feature has not been significantly updated since.

Similarly, the AutoQoS-Enterprise feature is based on the Cisco QoS Baseline document and the Enterprise QoS design guide (both published in 2002). Therefore it predates both RFC 4594 (published in 2006) and Cisco's Medianet Application Classes (published in 2008), as presented in Figure 1-25 of Chapter 1, "Enterprise Medianet Quality of Service Design 4.0—Overview." Beyond some minor differences in application class names, there are three notable differences between the AutoQoS-Enterprise application classes and the Cisco Medianet application classes:

- The AutoQoS-Enterprise feature does not support a Broadcast Video traffic class, which Cisco recommends to be marked as CS5 and which may be treated with an Expedite Forwarding PHB; this class can support applications such as live video broadcasts as well as IP video surveillance flows.
- The AutoQoS-Enterprise feature does not support a Realtime Interactive traffic class, which Cisco recommends to be marked as CS4 and may be treated with an Expedite Forwarding PHB; this class can support applications such as Cisco TelePresence.
- The AutoQoS-Enterprise feature marks Streaming Video traffic as CS4, whereas both RFC 4594 and Cisco recommend streaming media traffic to be classified and marked with an AF3 PHB; this class can support applications such as Cisco Digital Media System streams, particularly Video on Demand (VoD) flows.

Therefore, if AutoQoS-Enterprise is to be used to support medianet deployments, the administrator does well to keep these three key discrepancies in mind and accommodate as necessary. It should be noted that AutoQoS templates (both AutoQoS-VoIP and AutoQoS-Enterprise templates) can be customized to fit specific requirements, as needed.

Since the release of AutoQoS-Enterprise, new applications have been developed (such as IP video surveillance, TelePresence, and many others), additionally, new industry guidelines have since been published (such as RFC 4594). Furthermore, business requirements of QoS have continued to evolve. All of these factors combine to favor new QoS designs to support medianet applications, such as documented in this chapter, and are currently beyond what AutoQoS is able to support.

However, it should be noted that at the time of writing there are initiatives underway to upgrade AutoQoS functionality to match the current design recommendations for medianet as documented in this design chapter.

Control Plane Policing

Control plane policing (CPP) is an application of Quality of Service technologies in a security context that is available on switches and routers running Cisco IOS that allows the configuration of QoS policies that rate limit the traffic handled by the main CPU of the network device. This protects the control plane of the switch from direct DoS attacks and reconnaissance activity.

Note This section focuses on Control Plane Policing (CPP) implementation on Cisco IOS routers; Control Plane Policing (CoPP) for Cisco Catalyst 4500 and 6500 series switches is discussed in detail in Chapter 2, "Medianet Campus QoS Design 4.0."

CPP protects IOS-based routers by allowing the definition and enforcement of QoS policies that regulate the traffic processed by the main switch CPU (route or switch processor). With CPP, these QoS policies are configured to permit, block, or rate limit the packets handled by the main CPU.

A router can be logically divided into four functional components or planes:

- Data Plane
- Management Plane
- Control Plane
- Services Plane

The vast majority of traffic travels through the router via the data plane; however, the Route Processor (RP) must handle certain packets, such as routing updates, keepalives, and network management. This is often referred to as control and management plane traffic.

Because the Route Processor is critical to network operations, any service disruption to the Route Processor or the control and management planes can result in business-impacting network outages. A DoS attack targeting the Route Processor, which can be perpetrated either inadvertently or maliciously, typically involves high rates of punted traffic that result in excessive CPU utilization on the Route Processor itself. This type of attack, which can be devastating to network stability and availability, may display the following symptoms:

- High Route Processor CPU utilization (near 100%)
- Loss of line protocol keepalives and routing protocol updates, leading to route flaps and major network transitions
- Interactive sessions via the Command Line Interface (CLI) are slow or completely unresponsive due to high CPU utilization
- Route Processor resource exhaustion-resources such as memory and buffers are unavailable for legitimate IP data packets
- Packet queue backup, which leads to indiscriminate drops (or drops due to lack of buffer resources) of other incoming packets

CPP addresses the need to protect the control and management planes, ensuring routing stability, availability, and packet delivery. It uses a dedicated control-plane configuration via the Modular QoS CLI (MQC) to provide filtering and rate limiting capabilities for control plane packets.

Packets handled by the main CPU, referred to as control plane traffic, typically include:

- Routing protocols
- Packets destined to the local IP address of the router
- Packets from network management protocols, such as SNMP

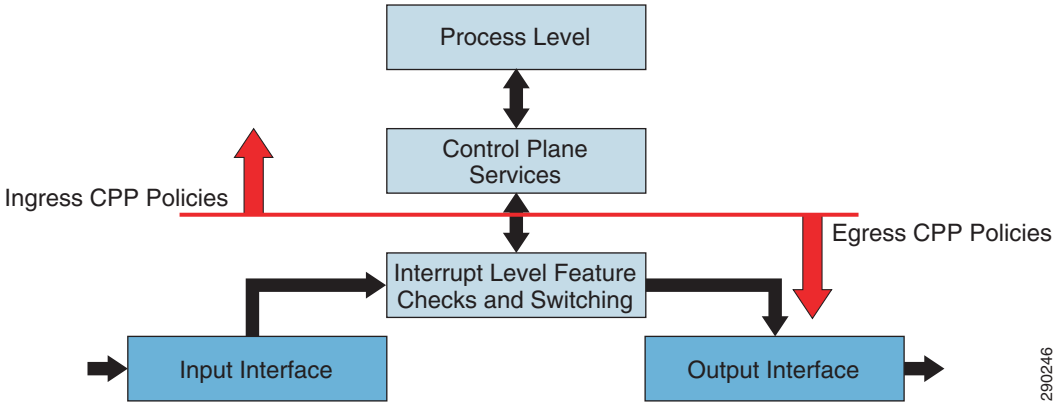
- Interactive access protocols, such as SSH, and telnet
- Other protocols, such as ICMP, or IP options, might also require handling by the switch CPU
- Layer 2 packets such as BPDU, CDP, DOT1X, etc.

CPP leverages the modular QoS command line interface (MQC) for its QoS policy configuration. MQC allows the classification of traffic into classes and lets you define and apply distinct QoS policies to separately rate limit the traffic in each class. MQC lets you divide the traffic destined to the CPU into multiple classes based on different criteria. For example, four traffic classes could be defined based on relative importance: critical, normal, undesirable, and default. After the traffic classes are defined, a QoS policy can be defined and enforced for each class according to importance. The QoS policies in each class can be configured to permit all packets, drop all packets, or drop only those packets exceeding a specific rate limit.

Note The number of control plane classes is not limited to four, but should be chosen based on local network requirements, security policies, and a thorough analysis of the baseline traffic.

CPP comes into play right after the routing decision and before traffic is forwarded to the control plane., as shown in Figure 3-19.

Figure 3-19 Cisco IOS Control Plane Policing (CPP) Operation



When CPP is enabled the sequence of events (at a high level) is:

1. A packet enters the router (configured with CPP) on the ingress interface.
2. The interface performs any applicable input port and QoS services.
3. The packet gets forwarded to the router CPU.
4. The CPU makes a routing or switching decision, determining whether or not the packet is destined for the control plane.
5. Packets destined for the control plane are processed by CPP and are dropped or delivered to the control plane according to each traffic class policy. Packets that have other destinations are forwarded normally.
6. (Optional) Packets originating from the control plane may also be subject to outbound CPP policies, to prevent the router from being utilized as a source of network reconnaissance or DoS attack.

CPP leverages MQC to define traffic classification criteria and to specify configurable policy actions for the classified traffic. Traffic of interest must first be identified via class-maps, which are used to define packets for a particular traffic class. Once classified, enforceable policy actions for the identified traffic are created with policy-maps. The control-plane global command allows the CP service policies to be attached to control plane itself, as will be detailed later in this design chapter.

Medianet WAN-Edge DiffServ QoS Model Design

Having reviewed these WAN QoS tools and considerations, it would be helpful at this point to also briefly review some tactical best-practices relating WAN edge QoS policy design, which will then help define optimal WAN edge policies. These tactical best-practices include

- Bandwidth Provisioning Guidelines for Realtime Traffic
- Bandwidth Provisioning Guidelines for Best Effort Traffic
- Bandwidth Provisioning Guidelines for Scavenger Traffic
- Guidelines for Enabling Fair-Queuing Pre-Sorters
- Guidelines for Enabling WRED
- WAN Edge Migration Models

Each of these best-practice guidelines will be reviewed in turn. Following these reviews, WAN edge models will be presented for 4, 8, and 12 class-of-service models.

Bandwidth Provisioning Guidelines for Realtime Traffic

The Realtime or strict priority class(es) of traffic corresponds to the RFC 3246 EF PHB. The amount of bandwidth assigned to the realtime queuing class is variable. However, if the majority of bandwidth is provisioned with strict priority queuing (which is effectively a FIFO queue), then the overall effect is a dampening of QoS functionality, both for latency and jitter sensitive realtime applications (contending with each other within the FIFO priority queue) and also for non-realtime applications (as these may periodically receive wild bandwidth allocation fluctuations, depending on the instantaneous amount of traffic being serviced by the priority queue). Remember the goal of convergence is to enable voice, video, and data applications to **transparently** co-exist on a single IP network. When realtime applications dominate a link, then non-realtime applications fluctuate significantly in their response times, destroying the transparency of the converged network.

For example, consider a (45 Mbps) T3/DS3 link configured to support two separate Cisco TelePresence 3000 systems. Each TelePresence call-operating at the highest quality setting-would require (at least) 15 Mbps of realtime traffic. Prior to these TelePresence calls being placed, non-realtime applications would have access to 100% of the bandwidth (to simplify the example, we are assuming there are no other realtime applications, such as VoIP, on this link). However, once these TelePresence calls are established, the realtime TelePresence calls would suddenly dominate more than 66% of the link and all non-realtime applications would just as suddenly be contending for less than 33% of the link. TCP windowing for many of these non-realtime applications would begin slow-starting, resulting in many data applications hanging, timing out, or becoming stuck in a non-responsive state. Such network behavior, changing from one minute to the next, generally translates into users calling the IT help desk complaining about the network (which happens to be functioning properly, albeit in a poorly-configured manner).

Cisco Technical Marketing has done extensive testing and has found that a significant decrease in non-realtime application response times occurs when realtime traffic exceeds one-third of link bandwidth capacity. Extensive testing and customer deployments have shown that a general best queuing

practice is to **limit the amount of strict priority queuing to 33% of link bandwidth capacity**. This strict priority queuing rule is a conservative and safe design ratio for merging realtime applications with data applications.

Note As previously discussed, Cisco IOS software allows the abstraction (and thus configuration) of multiple strict priority LLQs. In such a multiple LLQ context, this design principle would apply to the sum of all LLQs to be within one-third of link capacity.

It is vitally important to understand that **this strict priority queuing rule is simply a best practice design recommendation and is not a mandate**. There may be cases where specific business objectives cannot be met while holding to this recommendation. In such cases, enterprises must provision according to their detailed requirements and constraints. However, it is important to recognize the tradeoffs involved with over-provisioning strict priority traffic and its negative performance impact both on other realtime flows and also on non-realtime-application response times.

And finally, **any traffic assigned to a strict-priority queue should be governed by an admission control mechanism**.

Bandwidth Provisioning Guidelines for Best Effort Traffic

The Best Effort class is the default class for all traffic that has not been explicitly assigned to another application-class queue. Only if an application has been selected for preferential/deferential treatment is it removed from the default class. Because most enterprises have several thousand applications running over their networks, adequate bandwidth must be provisioned for this class as a whole in order to handle the sheer number and volume of applications that default to it. Therefore, it is recommended to **provision at least 25 percent of link bandwidth for the default Best Effort class**.

Bandwidth Provisioning Guidelines for Scavenger Traffic

Whenever Scavenger queuing class is enabled, it should be assigned a minimal amount of bandwidth, such as 1%.

Guidelines for Enabling Fair-Queuing Pre-Sorters

As of IOS 12.4(20)T, a **fair-queue** pre-softer can be applied to any CBWFQ class. A fair-queuing pre-sorter is very effective in managing TCP flows, as it prevents long-spanning TCP sessions from dominating the bandwidth for a given class (due to TCP's nature of continually expanding window sizes until maximum capacity is attained). Additionally, fair-queuing pre-sorters provide a measure of fairness also to UDP traffic classes (although UDP flows will not adjust their transmission rates as do TCP flows). However, enabling fair-queuing pre-sorters does not add much value when enabled on the control classes (of network, signaling or management traffic), as these time-sensitive flows should be serviced on a first-come-first-serve basis; furthermore, if control classes experience drops or excessive delays, then their respective queues servicing need to be expanded, not delayed further by introducing an additional round of queuing. Finally, enabling fair-queuing on the scavenger flows is optional, and since no service-level guarantee is offered to this traffic class an administrator may elect to save the marginal CPU cycles of adding an extra fair-queuing pre-sorter for this class.

To summarize, **it is recommended to enable fair-queuing pre-sorters on both TCP and UDP CBWFQ traffic classes, except for control classes and the scavenger class**.

Guidelines for Enabling WRED

Congestion avoidance mechanisms, such as WRED, work best with TCP-based applications because the selective dropping of packets causes the TCP windowing mechanisms to “throttle-back” and adjust the rate of flows to manageable rates. This avoids TCP-synchronization and results in more effective bandwidth utilization, both at a queuing-class level and at link-level. Therefore **it is recommended to enable WRED on all TCP-based traffic classes**.

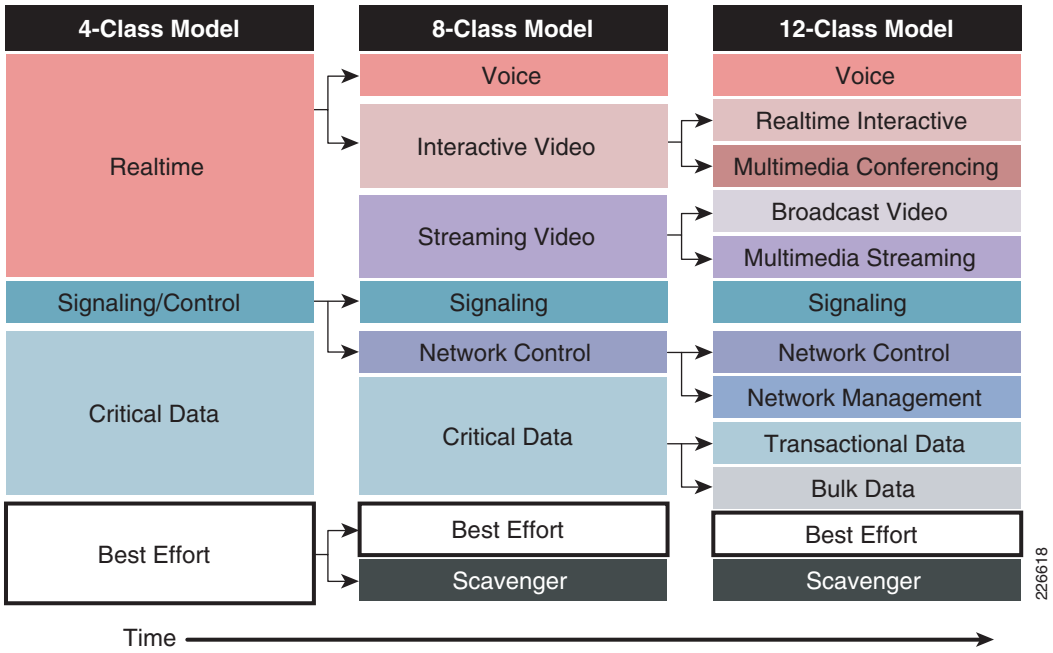
WRED is significantly less effective in managing congestion avoidance on UDP-based flows. However, it should be noted that today's multimedia conferencing and streaming applications aren't exclusively UDP-based anymore, but rather often have TCP flows interspersed among UDP flows to enable application sharing, chatting, file-transfers, etc. Therefore, **there is a degree of benefit obtained in managing congestion on these multimedia conferencing and streaming traffic classes also by enabling WRED**. This degree of benefit is increased if policing is enabled within the campus to increase drop precedence values of exceeding or violating flows.

Furthermore, as both RFC 4594 and Cisco’s medianet marking recommendations utilize Assured Forwarding Per-Hop Behaviors for the **multimedia conferencing, multimedia streaming, transactional data, and bulk data traffic classes** (marked AF4, AF3, AF2 and AF1, respectively), **each of these traffic classes should be enabled with DSCP-based WRED, as per the AF PHB standard (RFC 2597)**. For example, Drop Precedence 3 packets could be configured to begin randomly dropping at 60% of the queue-depth, Drop Precedence 2 packets at 70% and Drop Precedence 1 packets at 80%; all packets should be configured to tail-drop only at the tail of the queue (100%).

WAN Edge Migration Models

While RFC 4594 outlines a 12-class model, Cisco recognizes that not all enterprises are ready to implement such a complex QoS design. This be due to business reasons, technical constraints, or other reasons. Therefore, rather than considering the medianet QoS recommendations presented in this design chapter as an all-or-nothing approach, Cisco recommends considering a phased approach to media application class expansion, as illustrated in Figure 3-20.

Figure 3-20 Media Application Class Expansion Models

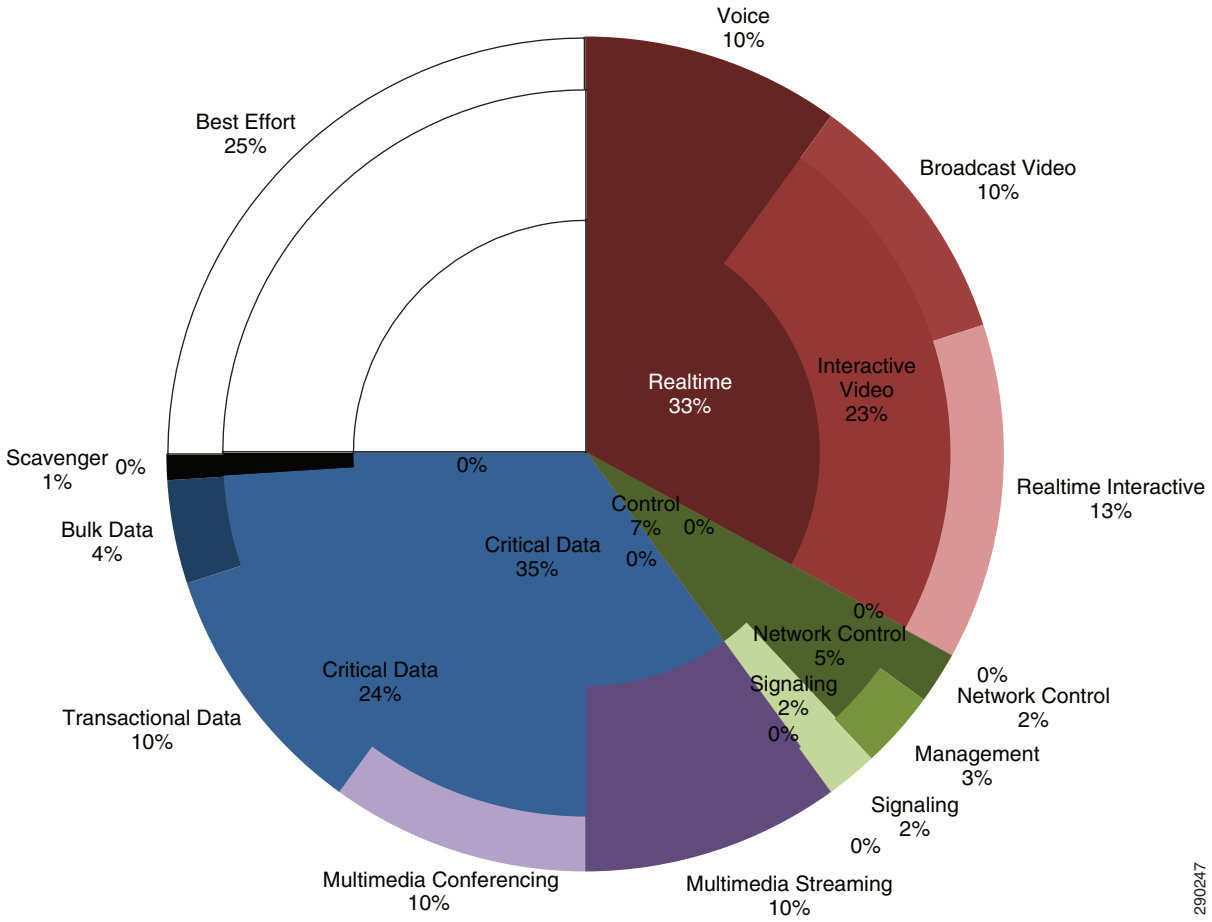


Utilizing such a phased approach to application class expansion, enterprise administrators can incrementally implement QoS policies across their infrastructures in a progressive manner, inline with their business needs and technical constraints. Familiarity with this enterprise medianet QoS model can assist in the smooth expansion of QoS policies to support additional media applications as future requirements arise. Nonetheless, at the time of QoS deployment, the enterprise needs to clearly define their business objectives with QoS, which correspondingly determines how many traffic classes will be required at each phase of deployment.

Marking codepoints should remain as consistent as possible as the number of application classes expand. Multiple application classes can be matched on a single codepoint or as a group of several codepoints, with the latter being the preferred option facilitating easier class-expansion over time. For example, in Figure 3-20 all realtime applications can be matched under a single class-map as a single codepoint (such as EF) to represent any and all realtime voice and video applications. Alternatively, these realtime applications may be matched under a single class-map using separate codepoints to identify discreet realtime applications (such as EF for Voice, CS5 for Broadcast Video and CS4 for Realtime Interactive). This second approach will make it easier to separate application classes down the road, as well as provide granular feedback as to the composition of the aggregate traffic class (as both the **show policy interface** verification command and the correspond **Class-Based QoS MIB** will report on traffic statistics for each **match** statement, giving the administrator a clearer picture of how much of his realtime traffic is voice versus video and so forth). Therefore, matching on multiple codepoints within a single aggregate class for simpler (4- or 8-class) WAN edge QoS models is the approach used in this design chapter.

Not only should marking values be consistent as application class-models expand, but also should bandwidth allocations remain consistent. To achieve this, administrators are encouraged to give forethought to how various traffic classes may need to be sub-divided over time, such that new classes will receive adequate levels of bandwidth without severely impacting user experiences of existing traffic classes. Figure 3-21 shows an example of how application class bandwidth allocations may be sub-divided over time into more granular QoS application-class models while retaining consistent bandwidth allocations; the innermost pie represents bandwidth allocations for a 4-class model, the intermediate pie represents an 8-class model, and the outermost pie represents a 12-class model.

Figure 3-21 Media Application Class Expansion Bandwidth Allocation Example



Having reviewed these considerations and tactical design recommendations, the individual WAN edge DiffServ models can now be defined on a per-class-model basis: one each for a 4-class model, an 8-class model and a 12-class model.

4-Class WAN Edge Model

In the 4-Class WAN Edge model the classes are: Realtime, Control, Critical-Data, and Best-Effort (which is represented as the default class).

The multiple-codepoint-per-class approach is used in this example to facilitate future class-expansion; however, administrators may choose to represent all applications within a given class as a single codepoint. Since a multiple-codepoint approach is used, it is critical to explicitly define the **class-map** logical operator to be **match-any** (which performs a logical OR operation on each **match** statement within the class-map), rather than the default logical operator, which is **match-all** (which represents a logical AND operation on each **match** statement within the class-map). A logical OR operation is required with the multi-codepoint approach, as DSCP codepoint values are mutually exclusive (i.e., a packet ***cannot*** be marked, for example, EF **AND** CS5 **AND** CS4 all at the same time; it must be one DSCP value **OR** another).

An additional note is that **match dscp** is used in all class-map examples in this design chapter, rather than **match ip dscp**. This is because **match ip dscp** is an older syntax which matches the DSCP values of only IPv4 packets, whereas **match dscp** will match the DSCP values of either IPv4 or IPv6 packets, thus future-proofing these designs to accommodate IPv6 also.

In this example, the Realtime class includes Voice (marked EF), Broadcast Video (marked CS5) and Realtime Interactive (marked CS4) and is aggregately provisioned with 33% LLQ. It should be noted that it is generally not a good practice to provision both voice and video applications within a single LLQ class, as video traffic is generally bursty and can interfere with voice; a dual-LLQ or multi-LLQ approach is recommended instead. However, with such a simplified class model, this configuration option doesn't really exist.

The Control class includes Network Control (marked CS6), Signaling (marked CS3) and Network Management (marked CS2), and is aggregately provisioned with 7% CBWFQ.

The Critical-Data class includes Multimedia-Conferencing (marked AF4), Multimedia-Streaming (marked AF3), Transactional-Data (marked AF2) and Bulk-Data (marked AF1), and is aggregately provisioned with 35% CBWFQ. A fair-queuing pre-sorter is enabled on the Critical-Data class, as is DSCP-based WRED. Incidentally, each AF class can be matched on a separate line (for example, **match af41 af42 af43**) or can be matched on three separate lines (**match af41**, **match af42**, and **match af43**); this is largely a matter of administrative preference, the only advantage of one approach over the other is that the latter will provide more visibility via the **show policy-map interface** verification command and the corresponding **Class-Based QoS MIB**. As this is a simplified 4-class model, the simpler approach is used in this instance, while the more complex approach will be used in the 12-class model that will follow.

Additionally, the WRED thresholds on the CRITICAL-DATA class may be optimized: By default the minimum WRED thresholds for each AF class are 24, 28 and 32 packets for Drop-Precedence values 3, 2, and 1 (respectively). These thresholds represent 60%, 70% and 80% (respectively) of the default queue-depth (of 64 packets). Also, by default the maximum WRED thresholds are set to 40 packets for all Drop-Precedence values for each AF class. Considering that the default queue-limit/depth is 64 packets, these default settings are inefficient on links experiencing sustained congestion that may cause a queue-depth in of 40 packets (at which point all codepoints will be tail-dropped, despite the queue having the capacity to accommodate another 24 packets). Thus, an administrator may choose to tune these WRED thresholds such that each AF class has a minimum WRED threshold of 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively), which represent (approximately) 60%, 70% and 80% of the default queue-depth of 64 packets and/or they may choose to tune the maximum WRED thresholds for each Drop-Precedence values for each AF class to the default queue-depth of 64 packets.

The default class is provisioned with an explicit bandwidth guarantee of 25% and has a fair-queuing pre-sorter enabled on it. Due to the sheer number of applications that are serviced by this default-queue, the queue-limit of this queue may be expanded beyond the default value (of 64 packets) to 128 packets (or even higher, per administrative preference). Additionally, WRED is recommended to be enabled on this default class, although there typically is only a single codepoint (of 0/DF) assigned to this queue; that is, unless Scavenger traffic has also been previously classified and marked, in which case there may be two codepoints assigned to this default class (0/DF and 8/CS1). In either case, the WRED thresholds are recommended to be tuned, as the default maximum and minimum thresholds for DSCP 0 are 20 and 40 packets, respectively; this would mean that packets marked with 0/DF would begin to be dropped at a packet-depth of 20 and would be tail-dropped at a packet-depth of 40 (despite the expanded queue-depth of 128 packets). The minimum drop-threshold for DSCP 0/DF could thus be tuned to 100 packets (approximately 80% of the queue-limit), with the maximum drop-threshold set to the tail of the queue (128 packets). Similarly, if Scavenger traffic is present, then the minimum drop-threshold for CS1 can be tuned to 60 with the maximum drop-threshold set to 100 (such that Scavenger traffic would be tail-dropped before any Best-Effort traffic would begin to be randomly-dropped).

The configuration for a 4-Class WAN Edge DiffServ model is shown in Example 3-3.

Example 3-3 A 4-Class WAN Edge DiffServ QoS Model Example

```
! This section defines the 4-Class class-maps
! (three are explicitly defined and one is default)
Router(config)# class-map match-any REALTIME
Router(config-cmap)# match dscp ef
 ! Matches VoIP
Router(config-cmap)# match dscp cs5
 ! Matches Broadcast Video
Router(config-cmap)# match dscp cs4
 ! Matches Realtime-Interactive

Router(config)# class-map match-any CONTROL
Router(config-cmap)# match dscp cs6
 ! Matches Network-Control
Router(config-cmap)# match dscp cs3
 ! Matches Signaling (control-plane traffic for voice/video infrastructure)
Router(config-cmap)# match dscp cs2
 ! Matches Network Management

Router(config)# class-map match-any CRITICAL-DATA
Router(config-cmap)# match dscp af41 af42 af43
 ! Matches Multimedia Conferencing on AF4
Router(config-cmap)# match dscp af31 af32 af33
 ! Matches Multimedia Streaming on AF3
Router(config-cmap)# match dscp af21 af22 af23
 ! Matches Transactional Data on AF2
Router(config-cmap)# match dscp af11 af12 af13
 ! Matches Bulk Data on AF1


! This section defines the 4-Class Policy-Map
Router(config)# policy-map WAN-EDGE-4-CLASS
Router(config-pmap)# class REALTIME
Router(config-pmap-c)# priority percent 33
 ! Provisions 33% LLQ for REALTIME class
Router(config-pmap-c)# class CONTROL
Router(config-pmap-c)#  bandwidth percent 7
 ! Provisions 7% CBWFQ for CONTROL class
Router(config-pmap-c)# class CRITICAL-DATA
Router(config-pmap-c)#  bandwidth percent 35
 ! Provisions 35% CBWFQ for CRITICAL-DATA class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on CRITICAL-DATA class
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on CRITICAL-DATA class
Router(config-pmap-c)#   random-detect dscp af11 50 64
 ! Tunes WRED min and max drop-thresholds for AF11/10 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af12 45 64
 ! Tunes WRED min and max drop-thresholds for AF12/12 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af13 40 64
 ! Tunes WRED min and max drop-thresholds for AF13/14 to 40 and 64 packets
Router(config-pmap-c)#   random-detect dscp af21 50 64
 ! Tunes WRED min and max drop-thresholds for AF21/18 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af22 45 64
 ! Tunes WRED min and max drop-thresholds for AF22/20 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af23 40 64
 ! Tunes WRED min and max drop-thresholds for AF23/22 to 40 and 64 packets
Router(config-pmap-c)#   random-detect dscp af31 50 64
 ! Tunes WRED min and max drop-thresholds for AF31/26 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af32 45 64
```

```
 ! Tunes WRED min and max drop-thresholds for AF32/28 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af33 40 64
 ! Tunes WRED min and max drop-thresholds for AF33/30 to 40 and 64 packets
Router(config-pmap-c)#   random-detect dscp af41 50 64
 ! Tunes WRED min and max drop-thresholds for AF41/34 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af42 45 64
 ! Tunes WRED min and max drop-thresholds for AF42/36 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af43 40 64
 ! Tunes WRED min and max drop-thresholds for AF43/38 to 40 and 64 packets
Router(config-pmap-c)# class class-default
Router(config-pmap-c)#  bandwidth percent 25
 ! Provisions 25% CBWFQ for default (Best-Effort) class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on default (Best-Effort) class
Router(config-pmap-c)#  queue-limit 128 packets
 ! Expands queue-limit to 128 packets
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on default (Best-Effort) class
Router(config-pmap-c)#   random-detect dscp default 100 128
 ! Tunes WRED min and max drop-thresholds for DF/0 to 100 and 128 packets
Router(config-pmap-c)#   random-detect dscp cs1 60 100
 ! Tunes WRED min and max drop-thresholds for CS1/8 to 60 and 100 packets
```

This configuration can be verified with the commands:

- **show class-map** (as shown in Example 3-4)
- **show policy-map** (as shown in Example 3-5)

Example 3-4 Verifying class-maps—show class-map

```
Router# show class-map
 Class Map match-any REALTIME (id 1)
   Match   dscp ef (46)
   Match   dscp cs5 (40)
   Match   dscp cs4 (32)

 Class Map match-any class-default (id 0)
   Match any

 Class Map match-any CRITICAL-DATA (id 3)
   Match   dscp af41 (34) af42 (36) af43 (38)
   Match   dscp af31 (26) af32 (28) af33 (30)
   Match   dscp af21 (18) af22 (20) af23 (22)
   Match   dscp af11 (10) af12 (12) af13 (14)

 Class Map match-any CONTROL (id 2)
   Match   dscp cs6 (48)
   Match   dscp cs3 (24)
   Match   dscp cs2 (16)

Router#
```

As can be seen in Example 3-4, each class-map is presented, with its corresponding logical operator (in this example, each class-map is using **match-any** as the logical operator), as well as each specific **match** statement criteria. A helpful feature of this verification command is that each DSCP keyword value is shown with its corresponding decimal value beside it, in parenthesis.

Example 3-5 Verifying policy-maps—show policy-map

```
Router# show policy-map
  Policy Map WAN-EDGE-4-CLASS
```

```
  Class REALTIME
    priority 33 (%)
  Class CONTROL
    bandwidth 7 (%)
  Class CRITICAL-DATA
    bandwidth 35 (%)
    fair-queue
     packet-based wred, exponential weight 9

    dscp    min-threshold    max-threshold    mark-probablity
    ------------------------------------------------------------
    af11 (10)     50               64               1/10
    af12 (12)     45               64               1/10
    af13 (14)     40               64               1/10
    af21 (18)     50               64               1/10
    af22 (20)     45               64               1/10
    af23 (22)     40               64               1/10
    af31 (26)     50               64               1/10
    af32 (28)     45               64               1/10
    af33 (30)     40               64               1/10
    af41 (34)     50               64               1/10
    af42 (36)     45               64               1/10
    af43 (38)     40               64               1/10
    default (0)   -                -                1/10
  Class class-default
    bandwidth 25 (%)
    fair-queue
    queue-limit 128 packets
     packet-based wred, exponential weight 9

    dscp    min-threshold    max-threshold    mark-probablity
    ------------------------------------------------------------
    cs1  (8)      60               100              1/10
    default (0)   100              128              1/10

Router#
```

As shown in Example 3-5, the **show policy-map** verification command confirms the key components of the QoS policy, including the LLQ/CBWFQ queuing strategy on a per-class basis (along with bandwidth percentages), whether or not a fair-queuing pre-sorter has been applied on a class, whether or not WRED has been applied on a class, the extended queue-limit of a given class, as well as any tuned DSCP-WRED drop-thresholds.

8-Class WAN Edge Model

In the 8-Class WAN Edge model the classes are: Voice, Interactive-Video, Network Control, Signaling, Multimedia Streaming, Critical Data, Scavenger and Best-Effort (which is represented as the default class).

The multiple-codepoint-per-class approach is used in this example for the Interactive-Video, Network Control, and Critical Data application classes; however, administrators may choose to represent all applications within a given class as a single codepoint. Whenever multiple-codepoints are used within a class (along with corresponding **match** statements) then the logical operator for the class-map must be **match-any**, and not the default **match-all**. However, if only a single-codepoint is being matched for a given class, then the default **match-all** logical operator works just the same; at which point the logical operator selected is simply a matter of administrative preference.

In this example, the dedicated Voice class matches on EF and is provisioned with 10% LLQ.

The Interactive-Video class matches includes both Broadcast Video (marked CS5) and Realtime Interactive (marked CS4) and is aggregately provisioned with 23% LLQ, via a dual-LLQ policy (one for Voice and another for Interactive-Video). Again, it should be noted that it is generally not a best-practice to provision multiple applications within a single LLQ class (in this case Broadcast Video and Realtime-Interactive video are sharing a single LLQ), as bursty video flows can interfere with each other. While this model is an improvement on the previous 4-class model (since now at least voice is protected from bursty video via implicit LLQ policers), this design is still sub-optimal, yet is a required trade-off inherent to this intermediately-complex application-class model.

The Network Control class includes Network Control (marked CS6) and Network Management (marked CS2), and is aggregately provisioned with 5% CBWFQ.

Signaling (marked CS3) is provisioned in a dedicated CBWFQ with a 2% bandwidth allocation.

Multimedia-Streaming (marked AF3) is provisioned in a dedicated CBWFQ with 10% bandwidth allocation. A fair-queuing pre-sorter is enabled on the Multimedia-Streaming class, as is DSCP-based WRED. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF3 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the AF3 maximum WRED thresholds for each Drop-Precedence value to the default queue-depth of 64 packets.

The Critical-Data class includes Multimedia-Conferencing (marked AF4), Transactional-Data (marked AF2) and Bulk-Data (marked AF1), and is aggregately provisioned with 24% CBWFQ. A fair-queuing pre-sorter is enabled on the Critical-Data class, as is DSCP-based WRED. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF4, AF2 and AF1 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the maximum WRED thresholds for each Drop-Precedence value for each AF group to the default queue-depth of 64 packets.

Scavenger traffic (marked CS1) is assigned a dedicated CBWFQ, which is bandwidth constrained to 1%. **It is important to note that to constrain the Scavenger class to 1 percent, an explicit bandwidth guarantee must be given to the Best-Effort class**. Otherwise, if class-default is not explicitly assigned a minimum bandwidth guarantee, the Scavenger class still can rob it of bandwidth. This is because of the way the CBWFQ algorithm has been coded: If classes protected with a **bandwidth** statement are offered more traffic than their minimum bandwidth guarantee, the algorithm tries to protect such excess traffic at the direct expense of robbing bandwidth from class-default (assuming that class-default is only configured with **fair-queue**), unless class-default itself also has a **bandwidth** statement (providing itself with a minimum bandwidth guarantee).

The default class is provisioned with an explicit bandwidth guarantee of 25% and has a fair-queuing pre-sorter enabled on it. Due to the sheer number of applications that are serviced by this default-queue, the queue-limit of this queue may be expanded beyond the default value (of 64 packets) to 128 packets (or even higher, per administrative preference). Additionally, WRED is recommended to be enabled on this default class. It is recommended to tune the WRED minimum drop-threshold for DSCP 0/DF to 100 packets (approximately 80% of the queue-limit), with the maximum drop-threshold set to the tail of the queue (128 packets).

The configuration for an 8-Class WAN Edge DiffServ model is shown in Example 3-6.

Example 3-6 An 8-Class WAN Edge DiffServ QoS Model Example

```
! This section defines the 8-Class class-maps
! (seven are explicitly defined and one is default)
Router(config)# class-map match-all VOICE
Router(config-cmap)# match dscp ef
 ! Matches VoIP

Router(config)# class-map match-any INTERACTIVE-VIDEO
```

```
Router(config-cmap)# match dscp cs5
 ! Matches Broadcast Video
Router(config-cmap)# match dscp cs4
 ! Matches Realtime-Interactive

Router(config)# class-map match-any NETWORK-CONTROL
Router(config-cmap)# match dscp cs6
 ! Matches Network Control
Router(config-cmap)# match dscp cs2
 ! Matches Network Management

Router(config)# class-map match-all SIGNALING
Router(config-cmap)# match dscp cs3
 ! Matches Signaling

Router(config)# class-map match-all MULTIMEDIA-STREAMING
Router(config-cmap)# match dscp af31 af32 af33
 ! Matches Multimedia-Streaming on AF3

Router(config)# class-map match-any CRITICAL-DATA
Router(config-cmap)# match dscp af41 af42 af43
 ! Matches Multimedia-Conferencing on AF4
Router(config-cmap)# match dscp af21 af22 af23
 ! Matches Transactional-Data on AF2
Router(config-cmap)# match dscp af11 af12 af13
 ! Matches Bulk-Data on AF1

Router(config)# class-map match-all SCAVENGER
Router(config-cmap)# match dscp cs1
 ! Matches Scavenger


! This section configures the 8-class policy-map
Router(config)# policy-map WAN-EDGE-8-CLASS
Router(config-pmap)# class VOICE
Router(config-pmap-c)#  priority percent 10
 ! Provisions 10% LLQ for VOICE class (dual LLQ-policy)
Router(config-pmap-c)# class INTERACTIVE-VIDEO
Router(config-pmap-c)#  priority percent 23
 ! Provisions 23% LLQ for INTERACTIVE-VIDEO class (dual-LLQ policy)
Router(config-pmap-c)# class NETWORK-CONTROL
Router(config-pmap-c)#  bandwidth percent 5
 ! Provisions 5% CBWFQ for NETWORK-CONTROL class
Router(config-pmap-c)# class SIGNALING
Router(config-pmap-c)#  bandwidth percent 2
 ! Provisions 2% CBWFQ for SIGNALING class
Router(config-pmap-c)# class MULTIMEDIA-STREAMING
Router(config-pmap-c)#  bandwidth percent 10
 ! Provisions 10% CBWFQ for MULTIMEDIA-STREAMING class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on MULTIMEDIA-STREAMING class
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on MULTIMEDIA-STREAMING class
Router(config-pmap-c)#   random-detect dscp af31 50 64
 ! Tunes WRED min and max drop-thresholds for AF31/26 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af32 45 64
 ! Tunes WRED min and max drop-thresholds for AF32/28 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af33 40 64
 ! Tunes WRED min and max drop-thresholds for AF33/30 to 40 and 64 packets
Router(config-pmap-c)# class CRITICAL-DATA
Router(config-pmap-c)#  bandwidth percent 24
 ! Provisions 24% CBWFQ for CRITICAL-DATA class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on CRITICAL-DATA class
```

```
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on CRITICAL-DATA class


Router(config-pmap-c)#   random-detect dscp af11 50 64
 ! Tunes WRED min and max drop-thresholds for AF11/10 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af12 45 64
 ! Tunes WRED min and max drop-thresholds for AF12/12 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af13 40 64
 ! Tunes WRED min and max drop-thresholds for AF13/14 to 40 and 64 packets
Router(config-pmap-c)#   random-detect dscp af21 50 64
 ! Tunes WRED min and max drop-thresholds for AF21/18 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af22 45 64
 ! Tunes WRED min and max drop-thresholds for AF22/20 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af23 40 64
 ! Tunes WRED min and max drop-thresholds for AF23/22 to 40 and 64 packets
Router(config-pmap-c)#   random-detect dscp af41 50 64
 ! Tunes WRED min and max drop-thresholds for AF41/34 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af42 45 64
 ! Tunes WRED min and max drop-thresholds for AF42/36 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af43 40 64
 ! Tunes WRED min and max drop-thresholds for AF43/38 to 40 and 64 packets
Router(config-pmap-c)# class SCAVENGER
Router(config-pmap-c)#  bandwidth percent 1
 ! Constrains Scavenger class to 1% CBWFQ
Router(config-pmap-c)# class class-default
Router(config-pmap-c)#  bandwidth percent 25
 ! Provisions 25% CBWFQ for default (Best-Effort) class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on default (Best-Effort) class
Router(config-pmap-c)#  queue-limit 128 packets
 ! Expands queue-limit to 128 packets
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on default (Best-Effort) class
Router(config-pmap-c)#   random-detect dscp default 100 128
 ! Tunes WRED min and max drop-thresholds for DF/0 to 100 and 128 packets
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

12-Class WAN Edge Model

In the 12-Class WAN Edge model the classes are: Voice, Broadcast-Video, Realtime-Interactive, Network Control, Signaling, Network Management, Multimedia-Conferencing, Multimedia Streaming, Transactional-Data, Bulk-Data, Scavenger, and Best-Effort (which is represented as the default class).

In this model, each application-class is assigned to its own dedicated queue and—for the most part—are represented by a single codepoint only; therefore, the default **class-map** logical operator of **match-all** can be used on most classes. The only exceptions are the Assured Forwarding classes, which-in this particular example-are being matching on each discreet Drop Precedence level (and as such require the **match-any** logical operator).

In this example, a multi-LLQ approach is utilized: the Voice class being matched on EF and is provisioned with 10% LLQ; the Broadcast Video class is matched on CS5 and is provisioned with 10% LLQ, and the Realtime-Interactive class is matched on CS4 and is provisioned with 13% LLQ. As each application class is provisioned with its own LLQ, each is also protected from any bursting from the other classes via implicit LLQ policers.

The Network Control (marked CS6) is provisioned with 2% CBWFQ. Signaling (marked CS3) is likewise provisioned with 2% CBWFQ. And Network Management (marked CS2) is provisioned with 3% CBWFQ.

Multimedia-Conferencing (marked AF4) is provisioned with 10% CBWFQ, along with a fair-queuing pre-sorter and DSCP-based WRED. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF4 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the AF4 maximum WRED thresholds for each Drop-Precedence value to the default queue-depth of 64 packets.

Multimedia-Streaming (marked AF3) is provisioned 10% CBWFQ, along with a fair-queuing pre-sorter and DSCP-based WRED. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF3 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the AF3 maximum WRED thresholds for each Drop-Precedence value to the default queue-depth of 64 packets.

Transactional-Data (marked AF2) is provisioned with 10% CBWFQ, along with a fair-queuing pre-sorter and DSCP-based WRED. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF2 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the AF2 maximum WRED thresholds for each Drop-Precedence value to the default queue-depth of 64 packets.

Bulk-Data (marked AF1) is constrained to 4% CBWFQ, but also has a fair-queuing pre-sorter and DSCP-based WRED enabled. An administrative option is the tuning of the WRED thresholds on this class: specifically, setting the AF1 minimum WRED thresholds to 40, 45 and 50 packets for Drop-Precedence values 3, 2, and 1 (respectively) and/or setting the AF1 maximum WRED thresholds for each Drop-Precedence value to the default queue-depth of 64 packets.

Note Each of the AF classes is matched using three individual **match** statements, one for each Drop Precedence value in this example. This approach increases administrative visibility into these traffic classes. This is the case because each individual **match** statement within a **class-map** will have its own dedicated counters tracking packets and bytes matched within the **show policy-map interface** verification command and the corresponding **Class-Based QoS MIB**. Thus, using this approach, the administrator can clearly see if any traffic from these Assured Forwarding traffic classes has been marked as conforming, exceeding or violating by any preceding single-rate or two-rate policers (such as described by RFC 2697 and RFC 2698, respectively).

Scavenger traffic (marked CS1) is assigned a dedicated CBWFQ, which is bandwidth constrained to 1%.

The default class is provisioned with an explicit bandwidth guarantee of 25% and has a fair-queuing pre-sorter enabled on it. Due to the sheer number of applications that are serviced by this default-queue, the queue-limit of this queue may be expanded beyond the default value (of 64 packets) to 128 packets (or even higher, per administrative preference). Additionally, WRED is recommended to be enabled on this default class. It is recommended to tune the WRED minimum drop-threshold for DSCP 0/DF to 100 packets (approximately 80% of the queue-limit), with the maximum drop-threshold set to the tail of the queue (128 packets).

The configuration for a 12-Class WAN Edge DiffServ model is shown in Example 3-7.

Example 3-7 A 12-Class WAN Edge DiffServ QoS Model Example

```
! This section configures the class-maps
! (eleven class maps are explicitly defined and one is default)
Router(config)# class-map match-all VOICE
Router(config-cmap)# match dscp ef
 ! Matches VoIP
```

```
Router(config)# class-map match-all BROADCAST-VIDEO
Router(config-cmap)# match dscp cs5
 ! Matches Broadcast Video

Router(config)# class-map match-all REALTIME-INTERACTIVE
Router(config-cmap)#  match dscp cs4
 ! Matches Realtime-Interactive

Router(config)# class-map match-all NETWORK-CONTROL
Router(config-cmap)# match dscp cs6
 ! Matches Network Control

Router(config)# class-map match-all SIGNALING
Router(config-cmap)# match dscp cs3
 ! Matches Signaling

Router(config)# class-map match-all NETWORK-MANAGEMENT
Router(config-cmap)# match dscp cs2
 ! Matches Network Management

Router(config)# class-map match-any MULTIMEDIA-CONFERENCING
Router(config-cmap)# match dscp af41
Router(config-cmap)# match dscp af42
Router(config-cmap)# match dscp af43
 ! Matches Multimedia-Conferencing
 ! One match statement per Drop Precedence increases visibility

Router(config)# class-map match-any MULTIMEDIA-STREAMING
Router(config-cmap)# match dscp af31
Router(config-cmap)# match dscp af32
Router(config-cmap)# match dscp af33
 ! Matches Multimedia-Streaming
 ! One match statement per Drop Precedence increases visibility

Router(config)# class-map match-any TRANSACTIONAL-DATA
Router(config-cmap)# match dscp af21
Router(config-cmap)# match dscp af22
Router(config-cmap)# match dscp af23
 ! Matches Transactional-Data
 ! One match statement per Drop Precedence increases visibility

Router(config)# class-map match-any BULK-DATA
Router(config-cmap)# match dscp af11
Router(config-cmap)# match dscp af12
Router(config-cmap)# match dscp af13
 ! Matches Bulk-Data
 ! One match statement per Drop Precedence increases visibility

Router(config)# class-map match-all SCAVENGER
Router(config-cmap)# match dscp cs1
 ! Matches Scavenger


! This section configures the 12-Class policy-map
Router(config)# policy-map WAN-EDGE-12-CLASS
Router(config-pmap)# class VOICE
Router(config-pmap-c)#  priority percent 10
 ! Provisions 10% LLQ to VOICE class (multi-LLQ policy)
Router(config-pmap-c)# class BROADCAST-VIDEO
Router(config-pmap-c)#  priority percent 10
 ! Provisions 10% LLQ to BROADCAST-VIDEO class (multi-LLQ policy)
Router(config-pmap-c)# class REALTIME-INTERACTIVE
Router(config-pmap-c)#  priority percent 13
```

```
! Provisions 13% LLQ to REALTIME-INTERACTIVE class (multi-LLQ policy)
Router(config-pmap-c)# class NETWORK-CONTROL
Router(config-pmap-c)#  bandwidth percent 2
 ! Provisions 2% CBWFQ to NETWORK-CONTROL class
Router(config-pmap-c)# class SIGNALING
Router(config-pmap-c)#  bandwidth percent 2
 ! Provisions 2% CBWFQ to SIGNALING class
Router(config-pmap-c)# class NETWORK-MANAGEMENT
Router(config-pmap-c)#  bandwidth percent 3
 ! Provisions 3% CBWFQ to NETWORK-MANAGEMENT class
Router(config-pmap-c)# class MULTIMEDIA-CONFERENCING
Router(config-pmap-c)#  bandwidth percent 10
 ! Provisions 10% CBWFQ to MULTIMEDIA-CONFERENCING class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on MULTIMEDIA-CONFERENCING class
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on MULTIMEDIA-CONFERENCING class
Router(config-pmap-c)#   random-detect dscp af41 50 64
 ! Tunes WRED min and max drop-thresholds for AF41/34 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af42 45 64
 ! Tunes WRED min and max drop-thresholds for AF42/36 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af43 40 64
 ! Tunes WRED min and max drop-thresholds for AF43/38 to 40 and 64 packets
Router(config-pmap-c)# class MULTIMEDIA-STREAMING
Router(config-pmap-c)#  bandwidth percent 10
 ! Provisions 10% CBWFQ to MULTIMEDIA-STREAMING class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on MULTIMEDIA-STREAMING class
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on MULTIMEDIA-STREAMING class
Router(config-pmap-c)#   random-detect dscp af31 50 64
 ! Tunes WRED min and max drop-thresholds for AF31/26 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af32 45 64
 ! Tunes WRED min and max drop-thresholds for AF32/28 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af33 40 64
 ! Tunes WRED min and max drop-thresholds for AF33/30 to 40 and 64 packets
Router(config-pmap-c)# class TRANSACTIONAL-DATA
Router(config-pmap-c)#  bandwidth percent 10
 ! Provisions 10% CBWFQ to TRANSACTIONAL-DATA class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on TRANSACTIONAL-DATA class
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on TRANSACTIONAL-DATA class
Router(config-pmap-c)#   random-detect dscp af21 50 64
 ! Tunes WRED min and max drop-thresholds for AF21/18 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af22 45 64
 ! Tunes WRED min and max drop-thresholds for AF22/20 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af23 40 64
 ! Tunes WRED min and max drop-thresholds for AF23/22 to 40 and 64 packets
Router(config-pmap-c)# class BULK-DATA
Router(config-pmap-c)#  bandwidth percent 4
 ! Provisions 4% CBWFQ to BULK-DATA class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on BULK-DATA class
Router(config-pmap-c)#   random-detect dscp-based
! Enables DSCP-based WRED on BULK-DATA class
Router(config-pmap-c)#   random-detect dscp af11 50 64
 ! Tunes WRED min and max drop-thresholds for AF11/10 to 50 and 64 packets
Router(config-pmap-c)#   random-detect dscp af12 45 64
 ! Tunes WRED min and max drop-thresholds for AF12/12 to 45 and 64 packets
Router(config-pmap-c)#   random-detect dscp af13 40 64
 ! Tunes WRED min and max drop-thresholds for AF13/14 to 40 and 64 packets
Router(config-pmap-c)# class SCAVENGER
Router(config-pmap-c)#  bandwidth percent 1
```

```
 ! Constrains Scavenger to 1% CBWFQ
Router(config-pmap-c)# class class-default
Router(config-pmap-c)#  bandwidth percent 25
 ! Provisions 25% CBWFQ for default (Best-Effort) class
Router(config-pmap-c)#  fair-queue
 ! Enables fair-queuing pre-sorter on default (Best-Effort) class
Router(config-pmap-c)#  queue-limit 128 packets
 ! Expands queue-limit to 128 packets
Router(config-pmap-c)#   random-detect dscp-based
 ! Enables DSCP-based WRED on default (Best-Effort) class
Router(config-pmap-c)#   random-detect dscp default 100 128
 ! Tunes WRED min and max drop-thresholds for DF/0 to 100 and 128 packets
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

SIP/SPA WAN Edge Policy Caveats and Amendments

QoS feature support on the SIP/SPA platforms can vary by SIP, by SPA, interface-type and by IOS version. The feature support matrix and restrictions relevant to the WAN edge policies presented in the previous sections are summarized below.

Note For full details on SIP/SPA QoS feature support, see: http://www.cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/6500series/76cfgsip.html#wp1177665.

When configuring queuing features—which include both congestion management and congestion avoidance features—on a SIP, consider the following guidelines:

- The SIPs and SPAs support different forms of queuing features, as shown in Table 3-4.
- The SIP-200 and SIP-400 do not support ingress queuing features.
- When configuring queuing on the SIP-400, consider the following guidelines:
  - A queue on the SIP-400 is not assured any minimum bandwidth.
  - You cannot configure bandwidth or shaping with queuing under the same class in a service policy on the SIP-400.
  - If you want to define bandwidth parameters under different classes in the same service policy on the SIP-400, then you only can use the **bandwidth remaining percent** command. The SIP-400 does not support other forms of the **bandwidth** command with queuing in the same service policy.
- You can use policing with queuing to limit the traffic rate.
- On the SIP-400, WRED is supported on bridged VCs with classification on precedence and DSCP values. On other SIPs, WRED does not work on bridged VCs (for example, VCs that implement MPB).
- When configuring WRED on the SIP-400, consider the following guidelines:
  - WRED is supported on bridged VCs with classification on precedence and DSCP values.

Table 3-4 provides information about which QoS queuing features are supported on the Cisco 7600 SIP-200, SIP-400, and SIP-600 (at the time of writing).

Table 3-4 QoS Congestion Management and Avoidance Features by SIP and SPA Combination

Congestion Management and Avoidance Feature	Cisco 7600 SIP-200	Cisco 7600 SIP-400	Cisco 7600 SIP-600[1]
Class-based Weighted Fair Queuing (CBWFQ) (**bandwidth** and **queue-limit** commands)	Supported for all SPAs.	Supported for all SPAs.	Supported for all SPAs.[1]
Flow-based Queuing (fair queueing) (**fair-queue** command)	Supported for all SPAs.	Not supported.	Not supported.
Low Latency Queuing (LLQ)/ Queueing (**priority** command)	Strict priority only—Supported for all SPAs.	Strict priority only—Supported for all SPAs.	Supported for all SPAs.[1]
Random Early Detection (RED) (**random-detect** commands)	Supported for all SPAs.	Supported for all SPAs. • ATM SPAs—Up to 106 unique WRED minimum threshold (min-th), maximum threshold (max-th), and mark probability profiles supported. • Other SPAs—Up to 128 unique WRED min-th, max-th, and mark probability profiles supported.	Not supported.
Weighted RED (WRED)	Supported for all SPAs, with the following exception: • WRED is not supported on bridged VCs.	Supported for all SPAs, with the following restriction: • WRED is supported on bridged VCs with classification on precedence and DSCP values.	Not supported.
Aggregate Weighted Random Early Detection (**random-detect aggregate**, **random-detect dscp (aggregate)**, and **random-detect precedence (aggregate)** commands)	Supported for ATM SPA PVCs only—Cisco IOS Release 12.2(18)SXE and later.	Supported for ATM SPA PVCs only—Cisco IOS Release 12.2(18)SXE and later.	Supported for all SPAs.[1]

1. Support for the Cisco 7600 SIP-600 was removed in Cisco IOS Release 12.2(33)SXH and restored in Cisco IOS Release 12.2(33)SXI and later releases.

The following examples demonstrate various IOS responses from SIP/SPA platforms to non-supported WAN edge QoS features, as well as their respective workarounds/policy-amendments.

Example 3-8 shows the SIP/SPA IOS response to interfaces not supporting priority percent.

Example 3-8 SIP/SPA WAN Edge Policy Error—priority percent command not supported

```
Router-SIP-SPA(config-if)# service-policy output WAN-EDGE
```

```
priority percent command is not supported for this interface
Configuration failed!
Router-SIP-SPA(config-if)#
```

In this case, IOS does not support the implicit policer that is normally included as a subcomponent of the LLQ feature. To work around this, first an explicit policer needs to be configured (to the same rate as the LLQ) and then a priority command needs to be configured (with no keywords). Functionally, the policies remain identical to regular LLQ policies.

In Example 3-9, a 4-Class WAN Edge policy is being applied to an OC3 ATM PVC on a SIP/SPA platform (which does not support **priority percent**), and as such requires an explicit policer to be defined prior to the use of the **priority** command.

Example 3-9 SIP/SPA WAN Edge Policy Amendment—explicit policing of LLQ

```
Router-SIP-SPA(config)# policy-map WAN-EDGE-4-CLASS-SIP-SPA
Router-SIP-SPA(config-pmap)# class REALTIME
Router-SIP-SPA(config-pmap-c)# police 50 mbps
 ! Defines an explicit policer to police LLQ traffic to 50 Mbps
Router-SIP-SPA(config-pmap-c-police)# priority
 ! Enables strict priority queuing on traffic admitted by the policer
…
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

Explicit policers can also be configured on dual- or multi-LLQ policies, as shown in Example 3-10, where a 12-Class WAN Edge policy has been amended to make use of multiple explicit policers.

Example 3-10 SIP/SPA WAN Edge Policy Amendment—explicit policing of multiple LLQ classes

```
Router-SIP-SPA(config)# policy-map WAN-EDGE-12-CLASS-SIP-SPA
Router-SIP-SPA(config-pmap)# class VOICE
Router-SIP-SPA(config-pmap-c)# police 15 mbps
 ! Defines an explicit policer to police LLQ traffic to 15 Mbps
Router-SIP-SPA(config-pmap-c-police)#  priority
 ! Enables strict priority queuing on traffic admitted by the policer
Router-SIP-SPA(config-pmap-c)# class BROADCAST-VIDEO
Router-SIP-SPA(config-pmap-c)#  police 15 mbps
 ! Defines an explicit policer to police LLQ traffic to 15 Mbps
Router-SIP-SPA(config-pmap-c-police)# priority
 ! Enables strict priority queuing on traffic admitted by the policer
Router-SIP-SPA(config-pmap-c)# class REALTIME-INTERACTIVE
Router-SIP-SPA(config-pmap-c)#  police 20 mbps
Router-SIP-SPA(config-pmap-c-police)# priority
 ! Enables strict priority queuing on traffic admitted by the policer
…
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

In turn, Example 3-11 shows the SIP/SPA IOS response to interfaces not supporting **fair-queue**.

Example 3-11 SIP/SPA WAN Edge Policy Error—fair-queue command not supported

```
Router-SIP-SPA(config-if)# service-policy output WAN-EDGE
```

```
fair-queue command is not supported for this interface
Configuration failed!
Router-SIP-SPA(config-if)#
```

The obvious first step to amend this WAN Edge policy is to remove any and all **fair-queue** statements, including the one on class-default. Additionally, **queue-limit** statements from the previous WAN EDGE policy-map examples should be removed from any SIP/SPA policies, as these platforms auto-configure their queue limits far in excess of the normal IOS default of 64 packets. For example, on the SIP-400, the default queue limit is calculated based on the number of 250-byte packets that the SIP can transmit in one half of a second. For example, for an OC-3 SPA with a rate of 155 Mbps, the default queue limit is 38,750 packets (155000000 x 0.5 / 250 x 8). That being the case, if WRED weights are to be tuned, then these will have to be tuned based on the IOS software-calculated SIP/SPA queue-limits.

In Example 3-12, a 4-Class WAN Edge policy is being amended in order to be applied on a POS interface on a SIP-400, which does not support **fair-queue** and also expands its queue-limits. The IOS software has calculated the queue-limit on the CRITICAL-DATA class to be 13,562 packets. As discussed in the previous sections, the AFx3 minimum WRED threshold can be tuned to 60% of the queue-depth (in this case approximately 8000 packets), the AFx2 minimum WRED threshold can be tuned to 70% of the queue-depth (in this case approximately 9000 packets) and the AFx3 minimum WRED threshold can be tuned to 80% of the queue-depth (in this case approximately 11,000 packets). Similarly, the IOS software has calculated the queue-depth of the default class to be 9687 packets in this example. Therefore the minimum WRED threshold for SCAVENGER (DSCP CS1/8) can be set to 60% (in this case approximately 6000 packets) and the maximum threshold can be set to 80% (in this case approximately 8000 packets). Following this, the minimum WRED threshold for default traffic (DSCP DF/0) can be set to 80% (in this case approximately 8000 packets) and the maximum threshold set to the queue limit (in this case 9687 packets).

Example 3-12 SIP/SPA WAN Edge Policy Amendment—removing fair-queue and tuning WRED

```
Router-SIP-SPA(config)# policy-map WAN-EDGE-4-CLASS-SIP-SPA
Router-SIP-SPA(config-pmap)# class REALTIME
Router-SIP-SPA(config-pmap-c)#  priority percent 33
Router-SIP-SPA(config-pmap-c)# class CONTROL
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 7
Router-SIP-SPA(config-pmap-c)# class CRITICAL-DATA
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 35
Router-SIP-SPA(config-pmap-c)#   random-detect dscp-based
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af11 11000 13562
 ! Tunes WRED min and max thresholds for AF11/10 to 11000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af12 9500 13562
 ! Tunes WRED min and max thresholds for AF12/12 to 9500 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af13 8000 13562
 ! Tunes WRED min and max thresholds for AF13/14 to 8000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af21 11000 13562
 ! Tunes WRED min and max thresholds for AF21/18 to 11000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af22 9500 13562
 ! Tunes WRED min and max thresholds for AF22/20 to 9500 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af23 8000 13562
 ! Tunes WRED min and max thresholds for AF23/22 to 8000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af31 11000 13562
 ! Tunes WRED min and max thresholds for AF31/26 to 11000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af32 9500 13562
 ! Tunes WRED min and max thresholds for AF32/28 to 9500 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af33 8000 13562
 ! Tunes WRED min and max thresholds for AF33/30 to 8000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af41 11000 13562
 ! Tunes WRED min and max thresholds for AF41/34 to 11000 and 13562 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af42 9500 13562
 ! Tunes WRED min and max thresholds for AF42/36 to 9500 and 13562 packets
```

```
Router-SIP-SPA(config-pmap-c)#   random-detect dscp af43 8000 13562
 ! Tunes WRED min and max thresholds for AF43/38 to 8000 and 13562 packets
Router-SIP-SPA(config-pmap-c)# class class-default
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 25
Router-SIP-SPA(config-pmap-c)#   random-detect dscp-based
Router-SIP-SPA(config-pmap-c)#   random-detect dscp cs1 6000 9687
 ! Tunes WRED min and max thresholds for CS1/8 to 8000 and 9687 packets
Router-SIP-SPA(config-pmap-c)#   random-detect dscp default 8000 9687
 ! Tunes WRED min and max thresholds for DF/0 to 8000 and 9687 packets
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

Finally, Example 3-13 shows the SIP/SPA IOS response to interfaces not supporting WRED (supporting aggregate WRED instead).

Example 3-13 SIP/SPA WAN Edge Policy Error—(non-aggregate) random-detect command not supported

```
Router-SIP-SPA(config-if)# service-policy output WAN-EDGE
random-detect w/o aggregate is not supported in output direction for this interface
Configuration failed!
Router-SIP-SPA(config-if)#
```

When configuring aggregate WRED on an ATM SPA interface, consider the following guidelines:

- The Aggregate WRED feature requires that the ATM SPAs are installed in a Cisco 7600 SIP-200 or a Cisco 7600 SIP-400.
- When you configure a policy map class for aggregated WRED on an ATM interface, then you cannot also configure the standard **random-detect** commands in interface configuration or policy-map class configuration mode.
- The set of subclass values defined on a **random-detect dscp-based aggregate** command will be aggregated into a single hardware WRED resource; the statistics for these subclasses will also be aggregated.
- Defining WRED parameter values for the default aggregate class is optional. If defined, WRED parameters applied to the default aggregate class will be used for all subclasses that have not been explicitly configured. If all possible DSCP values are defined as subclasses, a default specification is unnecessary. If the optional parameters for a default aggregate class are not defined and packets with an unconfigured IP precedence or DSCP value arrive at the interface, these undefined subclass values will be set based on interface (VC) bandwidth.
- After aggregate WRED has been configured in a service policy map, the service policy map must be applied at the ATM VC level
- The Aggregate WRED feature is not supported in a switched virtual circuit (SVC) environment.

Example 3-14 presents an amended 4-Class WAN Edge policy applied to an ATM PVC on a SIP-200. This example actually combines all three previous caveats: no support for LLQ implicit policing, no support for **fair-queue** and no support for WRED (only aggregate WRED). As such an explicit policer is defined for the LLQ, **fair-queue** statements are removed, as are **queue-limit** statements. The congestion avoidance policy has to be aggregate WRED, and as such the policy may only be applied directly to the ATM PVC. Additionally, The IOS software has calculated the queue-limit on the CRITICAL-DATA class to be 11,009 packets. As discussed in the previous sections, the AFx3 minimum WRED threshold can be tuned to 60% of the queue-depth (in this case approximately 6600 packets), the AFx2 minimum WRED threshold can be tuned to 70% of the queue-depth (in this case approximately

7700 packets) and the AFx3 minimum WRED threshold can be tuned to 80% of the queue-depth (in this case approximately 8800 packets). Similarly, the IOS software has calculated the queue-depth of the default class to be 9360 packets in this example. Therefore the minimum WRED threshold for SCAVENGER (DSCP CS1/8) can be set to 60% (in this case approximately 5600 packets) and the maximum threshold can be set to 80% (in this case approximately 7500 packets). Following this, the minimum WRED threshold for default traffic (DSCP DF/0) can be set to 80% (in this case approximately 7500 packets) and the maximum threshold set to the queue limit (in this case 9360 packets).

Example 3-14 SIP/SPA WAN Edge Policy Amendment—explicit policing of LLQ, removing fair-queue, and tuning aggregate WRED

```
Router-SIP-SPA(config)# policy-map WAN-EDGE-4-CLASS-SIP-SPA
Router-SIP-SPA(config-pmap)# class REALTIME
Router-SIP-SPA(config-pmap-c)# police 50 mbps
 ! Defines an explicit policer to police LLQ traffic to 50 Mbps
Router-SIP-SPA(config-pmap-c-police)# priority
 ! Enables strict priority queuing on traffic admitted by the policer
Router-SIP-SPA(config-pmap-c)# class CONTROL
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 6
Router-SIP-SPA(config-pmap-c)# class CRITICAL-DATA
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 35
Router-SIP-SPA(config-pmap-c)#  random-detect dscp-based aggregate
 ! Enables Aggregate DSCP-based WRED
Router-SIP-SPA(config-pmap-c)#  random-detect dscp values af13 af23 af33 af43
minimum-thresh 6600 maximum-thresh 11009
 ! Tunes WRED min-thresh for AFx3 to 6600 and max-thresh to 11009 packets
Router-SIP-SPA(config-pmap-c)#  random-detect dscp values af12 af22 af32 af42
minimum-thresh 7700 maximum-thresh 11009
 ! Tunes WRED min-thresh for AFx2 to 7700 and max-thresh to 11009 packets
Router-SIP-SPA(config-pmap-c)#  random-detect dscp values af11 af21 af31 af41
minimum-thresh 8800 maximum-thresh 11009
 ! Tunes WRED min-thresh for AFx1 to 8800 and max-thresh to 11009 packets
Router-SIP-SPA(config-pmap-c)# class class-default
Router-SIP-SPA(config-pmap-c)#  bandwidth percent 25
Router-SIP-SPA(config-pmap-c)#  random-detect dscp-based aggregate
 ! Enables Aggregate DSCP-based WRED
Router-SIP-SPA(config-pmap-c)#  random-detect dscp values cs1 minimum-thresh 5600
maximum-thresh 7500
 ! Tunes WRED min-thresh for CS1 to 5600 and max-thresh to 7500 packets
Router-SIP-SPA(config-pmap-c)#  random-detect dscp values default minimum-thresh 7500
maximum-thresh 9360
 ! Tunes WRED min-thresh for DF to 7500 and max-thresh to 9360 packets
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**

Link-Specific WAN-Edge DiffServ Design

The most popular private WAN media, at the time of writing, are leased-lines (serial links), ATM and POS. WAN edge designs for each of these link-specific media will be presented in turn.

Note This design chapter is primarily focused on private WAN designs. While Ethernet-based WAN handoffs are increasingly popular, these will be covered in a subsequent design chapter along with MPLS VPNs and Metro-Ethernet VPNs.

Serial Link/Leased-Line QoS Design

Serial links—sometimes referred to as leased-lines—may be encapsulated with High-Level Data-link Control Protocol (HDLC), Point-to-Point Protocol (PPP) or Multilink Point-to-Point Protocol (MLP). At lower link speeds (such as 64 kbps through ~5 Mbps) MLP offers the greatest flexibility, as it supports Link Fragmentation and Interleaving (LFI) for links ≤768 kbps, as well as supports the bundling of multiple physical links for efficient scaling; however, at medianet link speeds (T3/E3 minimum) there is no significant advantage of any of these three Layer 2 protocol options. As such, the Layer 2 encapsulation protocol is mainly matter of administrative preference; the IOS default Layer 2 encapsulation protocol on serial links is HDLC.

One consideration specific to T3 serial interfaces is that the Tx-Ring may need to be tuned. Cisco Technical Marketing has found that tuning the Tx-Ring to a value of 10 on T3 (and similarly on E3) serial interfaces is optimal for supporting HD-based rich media applications, such as Cisco TelePresence.

An optional consideration is tuning the load-interval. All interface rates are sampled over a period of time defined as the load-interval. By default, the load-interval will be 300 seconds (5 minutes). The load interval may range from 30 seconds to 600 seconds (10 minutes). To initially validate policy effectiveness, shorter load-intervals are recommended; thereafter, administrators may prefer to expand the load interval for longer-term trending statistics.

MQC policies are applied to all interfaces via the **service-policy** command; since these policies are egress queuing policies, the **output** keyword is required, as shown in Example 3-15.

Example 3-15 T3 Serial Link Service-Policy Application Example

```
! This section tunes the Tx-Ring & applies the MQC policy to a T3 serial int
interface Serial2/0
 description CAMPUS-TO-BRANCH-SERIAL-T3
 bandwidth 44210
 ip address 10.0.12.5 255.255.255.252
 load-interval 30
  ! Minimizes the interface-statistics sampling-period (optional)
 tx-ring-limit 10
  ! Optimizes the T3 Tx-Ring for medianet
 dsu bandwidth 44210
 framing c-bit
 cablelength 10
 serial restart-delay 0
 service-policy output WAN-EDGE-4-CLASS
  ! Attaches service policy to serial T3 interface
!
```

Note As previously mentioned, on IOS releases preceding HQF (which was introduced in IOS 12.4(20)T), the additional **max-reserved-bandwidth 100** interface command would also be required to support these medianet WAN-edge policies, as the total explicit bandwidth allocation of these policies is 100% (and as such exceed the legacy 75% explicit bandwidth allocation limit).

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**
- **show controllers** *interface x/y* **| include tx_limited** (as shown in Example 3-16)
- **show interface** (as shown in Example 3-17)

- **show policy-map interface** (as shown in Example 3-18)

Example 3-16 Verifying Tx-Ring Tuning—show controllers** interface x/y **| include tx_limited

```
Router# show controllers ser 2/0 | include tx_limited
tx_underrun_err=0, tx_soft_underrun_err=0, tx_limited=1(10)
```

As shown in Example 3-16, the Tx-Ring is confirmed to be tuned to a value of 10 packets, rather than the default value (for this interface) of 64.

Example 3-17 Verifying Interface Statistics—show interface

```
Router# show interface serial 2/0
Serial2/0 is up, line protocol is up
  Hardware is M1T-T3+ pa
  Description: CAMPUS-TO-BRANCH-SERIAL-T3
  Internet address is 10.0.12.5/30
  MTU 4470 bytes, BW 44210 Kbit/sec, DLY 200 usec,
     reliability 255/255, txload 255/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:09:58
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 754983
  Queueing strategy: Class-based queueing
  Output queue: 152/1000/0 (size/max total/drops)
  30 second input rate 1000 bits/sec, 3 packets/sec
  30 second output rate 44673000 bits/sec, 4766 packets/sec
     1911 packets input, 117730 bytes, 0 no buffer
     Received 70 broadcasts, 0 runts, 0 giants, 0 throttles
              0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     2851646 packets output, 3340730429 bytes, 0 underruns
     0 output errors, 0 applique, 0 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
    rxLOS inactive, rxLOF inactive, rxAIS inactive
    txAIS inactive, rxRAI inactive, txRAI inactive

Router#
```

Example 3-17 shows various interface-level statistics. Of note is the tx-load (transmit load) value of 255/255, indicating that the interface is fully-congested in the outbound direction. Because the interface is fully-congested (a requisite for queuing policies to engage), there are some corresponding output drops (in this case 754,983). The queuing strategy applied to this interface is Class-based queuing (which includes both LLQ and CBWFQ).

Example 3-18 Verifying Service-Policy Statistics—show policy-map interface (4-Class Policy-Map Example)

```
Router# show policy-map interface serial 2/0
 Serial2/0

  Service-policy output: WAN-EDGE-4-CLASS

    queue stats for all priority classes:

      queue limit 64 packets
```

```
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 1416378/1150100176

  Class-map: REALTIME (match-any)
    1416371 packets, 1150090152 bytes
    30 second offered rate 12992000 bps, drop rate 0 bps
    Match:  dscp ef (46)
      708188 packets, 135972096 bytes
      30 second rate 1536000 bps
    Match:  dscp cs5 (40)
      354091 packets, 507058312 bytes
      30 second rate 5728000 bps
    Match:  dscp cs4 (32)
      354092 packets, 507059744 bytes
      30 second rate 5728000 bps
    Priority: 33% (14589 kbps), burst bytes 364700, b/w exceed drops: 0


  Class-map: CONTROL (match-any)
    20055 packets, 27673370 bytes
    30 second offered rate 312000 bps, drop rate 0 bps
    Match:  dscp cs6 (48)
      763 packets, 47226 bytes
      30 second rate 0 bps
    Match:  dscp cs3 (24)
      7082 packets, 10141424 bytes
      30 second rate 114000 bps
    Match:  dscp cs2 (16)
      12210 packets, 17484720 bytes
      30 second rate 197000 bps
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops) 0/0/0
    (pkts output/bytes output) 20055/27674976
    bandwidth 7% (3094 kbps)


  Class-map: CRITICAL-DATA (match-any)
    1416375 packets, 2028249000 bytes
    30 second offered rate 22912000 bps, drop rate 4622000 bps
    Match:  dscp af41 (34) af42 (36) af43 (38)
      354093 packets, 507061176 bytes
      30 second rate 5728000 bps
    Match:  dscp af31 (26) af32 (28) af33 (30)
      354093 packets, 507061176 bytes
      30 second rate 5728000 bps
    Match:  dscp af21 (18) af22 (20) af23 (22)
      354095 packets, 507064040 bytes
      30 second rate 5728000 bps
    Match:  dscp af11 (10) af12 (12) af13 (14)
      354094 packets, 507062608 bytes
      30 second rate 5728000 bps
    Queueing
    queue limit 64 packets
    (queue depth/total drops/no-buffer drops/flowdrops) 59/285279/0/0
    (pkts output/bytes output) 1131114/1619755248
    bandwidth 35% (15473 kbps)
    Fair-queue: per-flow queue limit 16
      Exp-weight-constant: 9 (1/512)
      Mean queue depth: 65 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af11   150416/215395712   15041/21538712  21011/30087752    50      64    1/10
```

```
af12   101202/144921264   10120/14491840  12990/18601680    45      64     1/10
af13    34679/49660328     3468/4966176    9609/13760088    40      64     1/10
af21   150598/215656336   15060/21565920  20810/29799920    50      64     1/10
af22   101154/144852528   10115/14484680  13043/18677576    45      64     1/10
af23    34622/49578704     3463/4959016    9672/13850304    40      64     1/10
af31   150183/215062056   15019/21507208  21267/30454344    50      64     1/10
af32   100891/144475912   10090/14448880  13330/19088560    45      64     1/10
af33    35001/50121432     3501/5013432    9254/13251728    40      64     1/10
af41   150238/215140816   15024/21514368  21206/30366992    50      64     1/10
af42   101502/145350864   10151/14536232  12659/18127688    45      64     1/10
af43    34801/49835032     3480/4983360    9474/13566768    40      64     1/10


    Class-map: class-default (match-any)
      1416457 packets, 2028253172 bytes
      30 second offered rate 22912000 bps, drop rate 9853000 bps
      Match: any
      Queueing
      queue limit 128 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 97/608558/0/0
      (pkts output/bytes output) 807922/1156832159
      bandwidth 25% (11052 kbps)
      Fair-queue: per-flow queue limit 32
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 101 packets
dscp    Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
        pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

default 673833/964815303 43325/62040293       0/0          100     128   1/10
cs1     144212/206511584 14421/20650872  558440/799686080   60     100   1/10
Router#
```

As can be seen in Example 3-18, the **show policy-map interface** verification command provides the most visibility into MQC policies. Furthermore, all statistics gathered and displayed within the **show policy-map interface** command are also captured within the **Class-Based QoS MIB**, and may be walked or polled via SNMP. Specific to this configuration, the command verifies that there are no drops occurring on the REALTIME class nor on the CONTROL class. Additionally, while there are some drops on the CRITICAL-DATA class, these are being managed as efficiently as possible via the DSCP-based WRED policies on this class. Similarly, the default/Best-Effort class is also being managed effectively via DSCP-based WRED, such that Scavenger traffic is being dropped far more aggressively than default/Best-Effort traffic within this queuing-class.

ATM Link QoS Design

Asynchronous Transfer Mode (ATM) is one of the most flexible Layer 2 WAN encapsulation protocols. It can operate at fractional-T1 rates as well as at multi-gigabit line rates; it can be configured as a point-to-point protocol, as well as a multipoint protocol, due to its Non-Broadcast Multi-Access (NMBA) characteristics; furthermore, it can operate over copper or fiber.

As such, there are multiple ways to configure MQC service policies over ATM links. For example, Example 3-19 shows the configuration of service policies on the physical interface of an ATM link.

Example 3-19 ATM Link Service-Policy Application on the Physical Interface Example

```
! This section applies the MQC policy to an ATM physical interface
interface ATM4/0
 description CAMPUS-TO-BRANCH-ATM-OC3
 bandwidth 149760
```

```
 ip address 10.0.12.13 255.255.255.252
 load-interval 30
 no atm ilmi-keepalive
 no atm enable-ilmi-trap
 !
 service-policy output WAN-EDGE-8-CLASS
  ! Attaches service policy to the physical ATM interface
 pvc 0/112
 !
!
```

Alternatively, the MQC service policy may be attached to a *logical* interface, such as an ATM Permanent Virtual Circuit (PVC) defined under a sub-interface, as shown in Example 3-20.

Example 3-20 ATM Link Service-Policy Application on a Logical (ATM PVC) Interface Example

```
! This section applies the MQC policy to an ATM logical interface (PVC)
interface ATM4/0
 description CAMPUS-TO-BRANCH-ATM-OC3
 bandwidth 149760
 no ip address
 load-interval 30
 no atm ilmi-keepalive
 no atm enable-ilmi-trap
 !
!
interface ATM4/0.1 point-to-point
 description CAMPUS-TO-BRANCH-ATM-OC3-SUB-INT
 ip address 10.0.12.13 255.255.255.252
 no atm enable-ilmi-trap
 pvc 0/112
  description CAMPUS-TO-BRANCH-ATM-OC3-PVC
  vbr-rt 149760 149760
   ! Defines the ATM traffic contract: Variable Bit Rate - Realtime
  service-policy output WAN-EDGE-8-CLASS
   ! Attaches service policy to the physical ATM interface
 !
!
```

Which method is used is largely a matter of administrative preference, as well as what options the ATM carrier offers. If the option of defining policies under ATM PVCs is utilized, then one additional element needs to be defined: the ATM traffic contract of the PVC. ATM traffic contracts offer Layer 2 QoS on a per-PVC basis, and include options such as Constant Bit Rate (CBR), Variable-Bit Rate (VBR), Realtime (RT) and Non-Realtime (NRT). For example, Example 3-20 shows an ATM PVC with a Variable Bit Rate that is Realtime, as specified by the **vbr-rt** keywords. The specifics of what ATM PVC traffic contract is used isn't highly critical to the end-to-end QoS design, as MQC policies perform QoS at the packet-level (Layer 3), thus relegating ATM cell-level (Layer 2) QoS to a significant degree.

These configurations can be verified with the commands:

- **show class-map**
- **show policy-map**
- **show interface**
- **show atm pvc** (as shown in Example 3-21)
- **show policy-map interface** (as shown in Example 3-22)

Example 3-21 Verifying ATM PVC Statistics—show atm pvc interface

```
Router# show atm pvc 0/112
Description: CAMPUS-TO-BRANCH-ATM-OC3-PVC
ATM4/0.1: VCD: 1, VPI: 0, VCI: 112
VBR-RT, PeakRate: 149760 (353208 cps), Average Rate: 149760 (353208 cps)
Burst Cells: 94
AAL5-LLC/SNAP, etype:0x0, Flags: 0x4000040, VCmode: 0x0, Encapsize: 12
OAM frequency: 0 second(s), OAM retry frequency: 1 second(s)
OAM up retry count: 3, OAM down retry count: 5
OAM Loopback status: OAM Disabled
OAM VC Status: Not Managed
OAM Loop detection: Disabled
ILMI VC status: Not Managed
InARP frequency: 15 minutes(s)
Transmit priority 3
InPkts: 721, OutPkts: 4235151, InBytes: 47116, OutBytes: 3760494428
InPRoc: 253, OutPRoc: 1, Broadcasts: 50
InFast: 468, OutFast: 4235100, InAS: 0, OutAS: 0
InPktDrops: 0, OutPktDrops: 0/0/0 (holdq/outputq/total)
InByteDrops: 0, OutByteDrops: 0
CrcErrors: 0, SarTimeOuts: 0, OverSizedSDUs: 0, LengthViolation: 0, CPIErrors: 0
Out CLP=1 Pkts: 0
OAM cells received: 0
F5 InEndloop: 0, F5 InSegloop: 0, F5 InAIS: 0, F5 InRDI: 0
OAM cells sent: 0
F5 OutEndloop: 0, F5 OutSegloop: 0, F5 OutAIS: 0, F5 OutRDI: 0
OAM cell drops: 0
Status: UP

Router#
```

Example 3-21 displays various statistics per-ATM-PVC, including output packets and bytes. Additionally it confirms the ATM traffic contract, which in this case is VBR-RT.

Example 3-22 Verifying Service-Policy Statistics—show policy-map interface (8-Class Policy-Map Example)

```
Router# show policy-map interface atm 4/0.1 vc 0/112
 ATM4/0.1: VC 0/112 -

  Service-policy output: WAN-EDGE-8-CLASS

    queue stats for all priority classes:

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 5038733/2495362200

    Class-map: VOICE (match-all)
      3839004 packets, 767800800 bytes
      30 second offered rate 12800000 bps, drop rate 0 bps
      Match:  dscp ef (46)
      Priority: 10% (14976 kbps), burst bytes 374400, b/w exceed drops: 0


    Class-map: INTERACTIVE-VIDEO (match-any)
      1199687 packets, 1727549280 bytes
      30 second offered rate 28800000 bps, drop rate 0 bps
      Match:  dscp cs5 (40)
        479875 packets, 691020000 bytes
        30 second rate 11520000 bps
      Match:  dscp cs4 (32)
```

```
          719812 packets, 1036529280 bytes
          30 second rate 17280000 bps
        Priority: 23% (34444 kbps), burst bytes 861100, b/w exceed drops: 0


      Class-map: NETWORK-CONTROL (match-any)
        96079 packets, 138211488 bytes
        30 second offered rate 2304000 bps, drop rate 0 bps
        Match:  dscp cs6 (48)
          104 packets, 7488 bytes
          30 second rate 0 bps
        Match:  dscp cs2 (16)
          95975 packets, 138204000 bytes
          30 second rate 2304000 bps
        Queueing
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 0/0/0
        (pkts output/bytes output) 96079/138211488
        bandwidth 5% (7488 kbps)

      Class-map: SIGNALING (match-all)
        47987 packets, 69101280 bytes
        30 second offered rate 1152000 bps, drop rate 0 bps
        Match:  dscp cs3 (24)
        Queueing
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 0/0/0
        (pkts output/bytes output) 47988/69102720
        bandwidth 2% (2995 kbps)

      Class-map: MULTIMEDIA-STREAMING (match-all)
        719813 packets, 1036530720 bytes
        30 second offered rate 17280000 bps, drop rate 3262000 bps
        Match:  dscp af31 (26) af32 (28) af33 (30)
        Queueing
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops/flowdrops) 72/135899/0/0
        (pkts output/bytes output) 583915/840837600
        bandwidth 10% (14976 kbps)
        Fair-queue: per-flow queue limit 16
          Exp-weight-constant: 9 (1/512)
          Mean queue depth: 64 packets
  dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
         pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

  af31   343250/494280000  34325/49428000   47688/68670720   50      64    1/10
  af32   176041/253499040  17604/25349760   18988/27342720   45      64    1/10
  af33   77685/111866400    7768/11185920   12567/18096480   40      64    1/10

      Class-map: CRITICAL-DATA (match-any)
        2159444 packets, 3109599360 bytes
        30 second offered rate 51840000 bps, drop rate 18198000 bps
        Match:  dscp af41 (34) af42 (36) af43 (38)
          719812 packets, 1036529280 bytes
          30 second rate 17280000 bps
        Match:  dscp af21 (18) af22 (20) af23 (22)
          719813 packets, 1036530720 bytes
          30 second rate 17280000 bps
        Match:  dscp af11 (10) af12 (12) af13 (14)
          719819 packets, 1036539360 bytes
          30 second rate 17280000 bps
        Queueing
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops/flowdrops) 87/758108/0/0
```

```
      (pkts output/bytes output) 1401347/2017939680
      bandwidth 24% (35942 kbps)
      Fair-queue: per-flow queue limit 16
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 65 packets
dscp    Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
        pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af11   268113/386082720  26811/38607840   130339/187688160  50      64    1/10
af12   146415/210837600  14642/21084480    51575/74268000   45      64    1/10
af13    62719/90315360    6272/9031680     29033/41807520   40      64    1/10
af21   269276/387757440  26927/38774880   129060/185846400  50      64    1/10
af22   147428/212296320  14743/21229920    50460/72662400   45      64    1/10
af23    61805/88999200    6180/8899200     30038/43254720   40      64    1/10
af41   268108/386075520  26810/38606400   130346/187698240  50      64    1/10
af42   146191/210515040  14619/21051360    51821/74622240   45      64    1/10
af43    62626/90181440    6262/9017280     29135/41954400   40      64    1/10

    Class-map: SCAVENGER (match-all)
      479866 packets, 691007040 bytes
      30 second offered rate 11520000 bps, drop rate 10118000 bps
      Match:  dscp cs1 (8)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 64/421491/0
      (pkts output/bytes output) 58390/84081600
      bandwidth 1% (1497 kbps)

    Class-map: class-default (match-any)
      1559601 packets, 2245824032 bytes
      30 second offered rate 37440000 bps, drop rate 2395000 bps
      Match: any
      Queueing
      queue limit 128 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 114/99809/0/0
      (pkts output/bytes output) 1459812/2102127872
      bandwidth 25% (37440 kbps)
      Fair-queue: per-flow queue limit 32
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 103 packets
dscp    Transmitted         Random drop     Tail/Flow drop Minimum Maximum Mark
        pkts/bytes          pkts/bytes      pkts/bytes     thresh  thresh  prob

default 1492447/2149122272 102048/146949120  0/0            100     128    1/10
Router#
```

As can be seen in Example 3-22, the **show policy-map interface** command verifies that there are no drops occurring on any priority (LLQ) classes, which include the VOICE class and the INTERACTIVE-VIDEO class. Neither are there any drops on the NETWORK-CONTROL class and the SIGNALING class. Additionally, the command shows that DSCP-WRED is managing congestion in the MULTIMEDIA-STREAMING and CRITICAL-DATA class. Furthermore, the output displays that the Scavenger class is dropping aggressively, while DSCP-based WRED is controlling congestion in the default class, causing minimal tail-drops within it.

POS Link QoS Design

Packet over SONET (POS) is a communications protocol for transmitting packets in the form of Point to Point Protocol (PPP) over Synchronous Digital Hierarchy (SDH) or Synchronous Optical Networking (SONET), which are both standard protocols for communicating digital information using lasers or light

emitting diodes (LEDs) over optical fiber at high-speed line rates. POS is defined by RFC 2615 as PPP over SONET/SDH. PPP is the Point to Point Protocol that was designed as a standard method of communicating over point-to-point links. Since SONET/SDH utilizes point-to-point circuits, PPP is well suited for use over these links. Scrambling is performed during insertion of the PPP packets into the SONET/SDH frame. Even though it is possible to run PPP over SONET it is not a requirement. It is completely viable and often preferable to run the default HDLC on a Cisco POS interface, as HDLC carries much less overhead than PPP. HDLC encapsulation over POS can be verified with the **show interface pos** x/y verification command.

The basic unit of framing in SDH is a Synchronous Transport Module level 1 (STM-1), which operates at 155.52 Mbps. SONET refers to this basic unit as an Synchronous Transport Signal 3, concatenated (STS-3c), but its high-level functionality, frame size, and bit-rate are the same as STM-1. The relationship between Optical Carrier (OC) levels, SONET STS levels and SDH STM modules, along with payload and line rates is summarized in Table 3-5.

Table 3-5 OC/SONET/SDH Designations and Bandwidths

SONET Optical Carrier Level	SONET Frame Format	SDH level and Frame Format	Payload bandwidth (kbps[1])	Line Rate (kbps)
OC-1[2]	STS-1	STM-0	50,112	51,840
OC-3[3]	STS-3	STM-1	150,336	155,520
OC-12[4]	STS-12	STM-4	601,344	622,080
OC-24[5]	STS-24	–	1,202,688	1,244,160
OC-48[6]	STS-48	STM-16	2,405,376	2,488,320
OC-192[7]	STS-192	STM-64	9,621,504	9,953,280

1. http://en.wikipedia.org/wiki/Data_rate_units#Kilobit_per_second
2. http://en.wikipedia.org/wiki/OC-1
3. http://en.wikipedia.org/wiki/OC-3
4. http://en.wikipedia.org/wiki/OC-12
5. http://en.wikipedia.org/wiki/OC-24
6. http://en.wikipedia.org/wiki/OC-48
7. http://en.wikipedia.org/wiki/OC-192

POS interface configuration is quite simple, with MQC policies applied directly to the interface via the **service-policy** command, as shown in Example 3-23.

Example 3-23 POS Link Service-Policy Application Example

```
! This section applies the MQC policy to a POS physical interface
interface POS6/0
 description CAMPUS-TO-BRANCH-POS-OC3
 bandwidth 155000
 ip address 10.0.12.17 255.255.255.252
 load-interval 30
 !
 service-policy output WAN-EDGE-12-CLASS
  ! Attaches service policy to the physical POS interface
!
```

This configuration can be verified with the commands:

- **show class-map**

- **show policy-map**
- **show interface**
- **show policy-map interface** (as shown in Example 3-24)

Example 3-24 Verifying Service-Policy Statistics—show policy-map interface (12-Class Policy-Map Example)

```
Router# show policy-map interface pos 6/0
 POS6/0

  Service-policy output: WAN-EDGE-12-CLASS

    queue stats for all priority classes:

      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 4627164/2984695008

    Class-map: VOICE (match-all)
      2955876 packets, 567528192 bytes
      30 second offered rate 14147000 bps, drop rate 0 bps
      Match:  dscp ef (46)
      Priority: 10% (14976 kbps), burst bytes 374400, b/w exceed drops: 0


    Class-map: BROADCAST-VIDEO (match-all)
      738967 packets, 1058200744 bytes
      30 second offered rate 11457000 bps, drop rate 0 bps
      Match:  dscp cs5 (40)
      Priority: 10% (14976 kbps), burst bytes 374400, b/w exceed drops: 0


    Class-map: REALTIME-INTERACTIVE (match-all)
      923708 packets, 1322749856 bytes
      30 second offered rate 14322000 bps, drop rate 0 bps
      Match:  dscp cs4 (32)
      Priority: 13% (19468 kbps), burst bytes 486700, b/w exceed drops: 0


    Class-map: NETWORK-CONTROL (match-all)
      796 packets, 49092 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match:  dscp cs6 (48)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 796/50944
      bandwidth 2% (2995 kbps)

    Class-map: SIGNALING (match-all)
      73894 packets, 105816208 bytes
      30 second offered rate 1143000 bps, drop rate 0 bps
      Match:  dscp cs3 (24)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 75273/107790936
      bandwidth 2% (2995 kbps)

    Class-map: NETWORK-MANAGEMENT (match-all)
      147792 packets, 211638144 bytes
```

```
      30 second offered rate 2290000 bps, drop rate 0 bps
      Match:  dscp cs2 (16)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 149060/213453920
      bandwidth 3% (4492 kbps)

    Class-map: MULTIMEDIA-CONFERENCING (match-any)
      1477932 packets, 2116398624 bytes
      30 second offered rate 22914000 bps, drop rate 3862000 bps
      Match:  dscp af41 (34)
        640540 packets, 917253280 bytes
        30 second rate 9932000 bps
      Match:  dscp af42 (36)
        320270 packets, 458626640 bytes
        30 second rate 4967000 bps
      Match:  dscp af43 (38)
        517122 packets, 740518704 bytes
        30 second rate 8014000 bps
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 67/249090/0/0
      (pkts output/bytes output) 1228842/1759701744
      bandwidth 10% (14976 kbps)
      Fair-queue: per-flow queue limit 16
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 64 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af41   533437/763881784  53344/76388608   60142/86123344   50      64    1/10
af42   277738/397720816  27774/39772368   17949/25702968   45      64    1/10
af43   429912/615633984  42992/61564544   49375/70705000   40      64    1/10

    Class-map: MULTIMEDIA-STREAMING (match-any)
      1477935 packets, 2116402920 bytes
      30 second offered rate 22915000 bps, drop rate 3861000 bps
      Match:  dscp af31 (26)
        640539 packets, 917251848 bytes
        30 second rate 9931000 bps
      Match:  dscp af32 (28)
        320271 packets, 458628072 bytes
        30 second rate 4967000 bps
      Match:  dscp af33 (30)
        517125 packets, 740523000 bytes
        30 second rate 8016000 bps
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 71/249093/0/0
      (pkts output/bytes output) 1228842/1759701744
      bandwidth 10% (14976 kbps)
      Fair-queue: per-flow queue limit 16
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 64 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af31   534072/764791104  53407/76478824   59443/85122376   50      64    1/10
af32   278117/398263544  27812/39826784   17532/25105824   45      64    1/10
af33   428903/614189096  42890/61418480   50486/72295952   40      64    1/10

    Class-map: TRANSACTIONAL-DATA (match-any)
      1477932 packets, 2116398624 bytes
```

```
      30 second offered rate 22915000 bps, drop rate 3856000 bps
      Match:  dscp af21 (18)
        640540 packets, 917253280 bytes
        30 second rate 9932000 bps
      Match:  dscp af22 (20)
        320269 packets, 458625208 bytes
        30 second rate 4967000 bps
      Match:  dscp af23 (22)
        517123 packets, 740520136 bytes
        30 second rate 8016000 bps
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 65/249097/0/0
      (pkts output/bytes output) 1228835/1759691720
      bandwidth 10% (14976 kbps)
      Fair-queue: per-flow queue limit 16
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 65 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af21   533571/764073672  53357/76407224   59995/85912840   50      64    1/10
af22   277641/397581912  27764/39758048   18056/25856192   45      64    1/10
af23   429877/615583864  42988/61558816   49409/70753688   40      64    1/10

    Class-map: BULK-DATA (match-any)
      2216901 packets, 3174602232 bytes
      30 second offered rate 34373000 bps, drop rate 26751000 bps
      Match:  dscp af11 (10)
        640541 packets, 917254712 bytes
        30 second rate 9931000 bps
      Match:  dscp af12 (12)
        320270 packets, 458626640 bytes
        30 second rate 4966000 bps
      Match:  dscp af13 (14)
        1256090 packets, 1798720880 bytes
        30 second rate 19475000 bps
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops/flowdrops) 69/1725401/0/0
      (pkts output/bytes output) 491500/703828000
      bandwidth 4% (5990 kbps)
      Fair-queue: per-flow queue limit 16
        Exp-weight-constant: 9 (1/512)
        Mean queue depth: 65 packets
dscp   Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
       pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

af11   139388/199603616  13938/19959216   493599/706833768  50     64    1/10
af12    90683/129858056   9068/12985376   223710/320352720  45     64    1/10
af13   266331/381385992  26633/38138456   975647/1397126504 40     64    1/10

    Class-map: SCAVENGER (match-all)
      738968 packets, 1058202176 bytes
      30 second offered rate 11459000 bps, drop rate 9552000 bps
      Match:  dscp cs1 (8)
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 64/616084/0
      (pkts output/bytes output) 122884/175969888
      bandwidth 1% (1497 kbps)

    Class-map: class-default (match-any)
      3325441 packets, 4761914000 bytes
```

```
        30 second offered rate 51560000 bps, drop rate 3921000 bps
        Match: any
        Queueing
        queue limit 128 packets
        (queue depth/total drops/no-buffer drops/flowdrops) 118/253259/0/0
        (pkts output/bytes output) 3072182/4399248222
        bandwidth 25% (37440 kbps)
        Fair-queue: per-flow queue limit 32
          Exp-weight-constant: 9 (1/512)
          Mean queue depth: 105 packets
dscp    Transmitted       Random drop      Tail/Flow drop Minimum Maximum Mark
        pkts/bytes        pkts/bytes       pkts/bytes     thresh  thresh  prob

default 3102802/4443096062 255783/366280146  0/0           100     128   1/10
Router#
```

As shown in Example 3-24, the **show policy-map interface** command verifies that there are no drops occurring on any priority (LLQ) class, which includes the VOICE class, the BROADCAST-VIDEO class, and the REALTIME-INTERACTIVE class. Neither are there any drops on the NETWORK-CONTROL class and the SIGNALING class. Additionally, the command shows that DSCP-WRED is managing congestion in the MULTIMEDIA-CONFERENCING, MULTIMEDIA-STREAMING, TRANSACTIONA-DATA and BULK-DATA classes. Furthermore, the output displays that the Scavenger class is dropping aggressively, while DSCP-based WRED is controlling congestion in the default class, causing minimal tail-drops within it.

RSVP Design

Before RSVP can be effectively enabled on a per-interface basis, the IOS RSVP Agent functionality must be enabled on a router. The Cisco RSVP Agent is a Cisco IOS feature that enables Cisco Unified CallManager (CUCM) to perform RSVP-based call admission control. The Cisco RSVP Agent feature was introduced in Cisco IOS Release 12.4(6)T. Detailed configuration guidance and best practices for the IOS RSVP Agent is beyond the scope of this chapter, as it is very CUCM-oriented. Design guidance for IOS RSVP Agent is detailed in the Cisco Unified Communications System 8.x SRND in Chapter 11 at: http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/srnd/8x/cac.html.

Once the IOS RSVP Agent has been properly configured to communicate with the CUCM to enable admission control for voice and/or video policies, then the administrator can focus on interface-specific RSVP design. Two RSVP models will now be presented: a basic RSVP model and a more advanced model that features Application ID functionality.

Basic RSVP Design (IntServ/DiffServ Model)

Once the IOS RSVP Agent has been configured, then basic RSVP functionality is enabled by the **ip rsvp bandwidth** interface configuration command, which specifies how much bandwidth may be explicitly reserved by RSVP requests (the default is 75% of a link's bandwidth).

If all the PQ traffic is RSVP-enabled, the value specified in the **ip rsvp bandwidth** command and the **priority** LLQ command should match once Layer 2 overhead of the priority queue bandwidth has been taken into account. But if some PQ traffic is not RSVP-enabled, then ensure that the sum of the values specified in the **ip rsvp bandwidth** command and in the out-of-band call admission control mechanism do not exceed the bandwidth value specified in the **priority** command.

Also, if RSVP is enabled on one or more interfaces of a router, all interfaces through which you expect RSVP signaling to transit should also be enabled for RSVP to ensure that RSVP messages do not get dropped.

RSVP should be enabled at the edge of the network, including the router WAN interfaces on both sides of the WAN link, at all possible WAN congestion points, including redundant links of different speeds.

When deploying RSVP in the IP WAN in it is recommended to utilize the IntServ/DiffServ model, such that RSVP's only application to perform admission control, while DiffServ policies will perform classification, marking, policing and scheduling. This model scales much better-as RSVP does not have to maintain state of all flows-and also better integrates with the DiffServ policies presented earlier in this design chapter. Configuring the IntServ/DiffServ model requires two additional interface configuration commands: **ip rsvp resource provider none** and **ip rsvp data-packet classification none**. These two additional commands instruct RSVP not to perform QoS operations that are handled within the data plane by DiffServ policies.

Also it is recommended to use the **ip rsvp signaling dscp** interface configuration command to ensure that RSVP packets are marked to CS3/24, inline with other signaling flows (as by default these packets will be marked to 63).

A basic IntServ/DiffServ RSVP model is shown in Example 3-25.

Example 3-25 Basic IntServ/DiffServ RSVP Model Configuration

```
! This section configures basic IntServ/DiffServ RSVP on a WAN interface
interface Serial2/0
 description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
 bandwidth 44210
 ip address 10.0.12.5 255.255.255.252
 load-interval 30
  ! Minimizes the interface-statistics sampling-period (optional)
 tx-ring-limit 10
  ! Optimizes the T3 Tx-Ring for medianet
 dsu bandwidth 44210
 framing c-bit
 cablelength 10
 serial restart-delay 0
!
 service-policy output WAN-EDGE-DIFFSERV-POLICY
  ! Attaches the DiffServ MQC policy to the interface
 ip rsvp bandwidth 15000
  ! Specifies the amount of reservable BW (should match LLQ BW)
 ip rsvp signalling dscp 24
  ! Marks RSVP signaling traffic to CS3
 ip rsvp data-packet classification none
  ! Enables the IntServ/DiffServ model by disabling RSVP for classification
 ip rsvp resource-provider none
  ! Enables the IntServ/DiffServ model by disabling RSVP for scheduling
!
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**
- **show interface**
- **show policy-map interface**
- **show ip rsvp interface** (as shown in Example 3-26)
- **show ip rsvp interface detail** (as shown in Example 3-27)

- **show ip rsvp installed** (as shown in Example 3-28)

Note While there are an array of **show ip rsvp** verification commands, these are the best suited to the context of this design example.

Example 3-26 Verifying a Basic IntServ/DiffServ RSVP Model—show ip rsvp interface

```
Router# show ip rsvp interface serial 2/0
interface    rsvp  allocated  i/f max  flow max sub max  VRF
Se2/0        ena   9996k           15M      15M      0
Router#
```

As shown in Example 3-26, this interface has been allocated 15 Mbps of bandwidth that may be reserved via RSVP, of which currently 9996 kbps is being allocated.

Example 3-27 Verifying a Basic IntServ/DiffServ RSVP Model—show ip rsvp interface detail

```
Router# show ip rsvp interface detail serial 2/0

 Se2/0:
   RSVP: Enabled
   Interface State: Up
   Bandwidth:
     Curr allocated: 9996k bits/sec
     Max. allowed (total): 15M bits/sec
     Max. allowed (per flow): 15M bits/sec
     Max. allowed for LSP tunnels using sub-pools: 0 bits/sec
     Set aside by policy (total): 0 bits/sec
   Admission Control:
     Header Compression methods supported:
       rtp (36 bytes-saved), udp (20 bytes-saved)
   Traffic Control:
     RSVP Data Packet Classification is OFF
     RSVP resource provider is: none
   Signalling:
     DSCP value used in RSVP msgs: 0x18
     Number of refresh intervals to enforce blockade state: 4
   Authentication: disabled
     Key chain:   <none>
     Type:        md5
     Window size: 1
     Challenge:   disabled
   Hello Extension:
     State: Disabled
Router#
```

Example 3-27 provides more detail regarding the basic RSVP policy. For example, it verifies that 9996 kbps of a maximum of 15 Mbps is currently being reserved and allocated to media flows. Also it confirms that there are no data classification nor resource provider functions being performed by RSVP (because RSVP is operating in the IntServ/DiffServ model). And finally that RSVP packets are being marked to CS3/24 (which is expressed in hexadecimal as 0x18).

Example 3-28 Verifying a Basic IntServ/DiffServ RSVP Model—show ip rsvp installed

```
Router# show ip rsvp installed
80K    10.17.100.101    10.16.255.101   UDP    3125726645
80K    10.17.100.102    10.16.255.102   UDP    1677321718
80K    10.17.100.103    10.16.255.103   UDP    2647016878
```

```
80K    10.17.100.104    10.16.255.104   UDP    1811517845
80K    10.17.100.105    10.16.255.105   UDP    2529422658
80K    10.17.100.106    10.16.255.106   UDP    2461326904
80K    10.17.100.107    10.16.255.107   UDP    2317526733
80K    10.17.100.108    10.16.255.108   UDP    2213923492
80K    10.17.100.109    10.16.255.109   UDP    2559029755
80K    10.17.100.110    10.16.255.110   UDP    2785432460
80K    10.17.100.111    10.16.255.111   UDP    2418929516
80K    10.17.100.112    10.16.255.112   UDP    2239517433
753K   10.17.100.101    10.16.255.101   UDP    2150228165
753K   10.17.100.102    10.16.255.102   UDP    1778130069
753K   10.17.100.103    10.16.255.103   UDP    2991825300
753K   10.17.100.104    10.16.255.104   UDP    2970728284
753K   10.17.100.105    10.16.255.105   UDP    1778019823
753K   10.17.100.106    10.16.255.106   UDP    2830120751
753K   10.17.100.107    10.16.255.107   UDP    3123019925
753K   10.17.100.108    10.16.255.108   UDP    2006732170
753K   10.17.100.109    10.16.255.109   UDP    3067717868
753K   10.17.100.110    10.16.255.110   UDP    2228120874
753K   10.17.100.111    10.16.255.111   UDP    2405616900
753K   10.17.100.112    10.16.255.112   UDP    2220332146

Router#
```

Example 3-28 breaks down the composition of the 9996 kbps of bandwidth that is currently being reserved and allocated to media flows, showing that there are a dozen voice calls (at 80 kbps each) and a dozen video calls (at 753 kbps each) that are currently active across the interface.

Advanced RSVP Design (IntServ/DiffServ Model with Local Policies and Application IDs)

The RSVP Local Policy provides the mechanism for controlling a reservation based on an Application ID. Application IDs are mapped to RSVP Local Policies through the **ip rsvp policy identity** command. RSVP Local Policy identities are defined globally and are available to each interface for policy enforcement. Each identity can have one policy locator defined to match an Application ID.

To give the user as much flexibility as possible in matching application policy locators to local policies, the RSVP local policy command line interface (CLI) accepts application ID match criteria in the form of Unix-style regular expressions for the policy locator. Regular expressions are already used in the CLI for existing Cisco IOS components such as Border Gateway Protocol (BGP).

Note Refer to the follow documentation for more information on how regular expressions are used in Cisco IOS: Using Regular Expressions in BGP http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094a92.shtml.

As previously mentioned, CUCM can set RFC 2872 Application IDs for both voice and video flows as cluster-wide service parameters. By default, the Application IDs used by CUCM are “AudioStream” for voice flows and “VideoStream” for video flows (for both audio and video components).

Therefore, the globally-defined IOS RSVP policy identities that match these CUCM Application IDs for voice and video are (respectively) are:

ip rsvp policy identity rsvp-voice policy-locator .*AudioStream.*

ip rsvp policy identity rsvp-video policy-locator .*VideoStream.*

In turn, local policies based on Application IDs are applied to an interface using the **ip rsvp policy local identity** command. For reservations that match its policy locator value, a local policy has the ability to perform the following functions:

- Define the maximum amount of bandwidth the reservations can reserve as a group or as a single sender.
- Forward or not forward RSVP messages.
- Accept or not accept RSVP messages.
- Define the maximum bandwidth the group or sender can reserve.

There is also a catch-all local policy called the default local policy. This local policy will match any RSVP reservation that did not match the other RSVP local policies configured on the link. The default local policy can be used to match reservations that are not tagged with an Application ID or reservations that are tagged with an Application ID that is to be treated as untagged traffic.

An advanced RSVP model featuring local policies for voice and video application IDs is shown in Example 3-29.

Example 3-29 Advanced Local Policy with Application ID RSVP Model Configuration

```
! This section configures advanced AppID RSVP on a WAN interface
interface Serial2/0
 description CAMPUS-TO-BRANCH-SERIAL-T3-WITH-RSVP
 bandwidth 44210
 ip address 10.0.12.5 255.255.255.252
 load-interval 30
  ! Minimizes the interface-statistics sampling-period (optional)
 tx-ring-limit 10
  ! Optimizes the T3 Tx-Ring for medianet
 dsu bandwidth 44210
 framing c-bit
 cablelength 10
 serial restart-delay 0
!
 service-policy output WAN-EDGE-DIFFSERV-POLICY
  ! Attaches the DiffServ MQC policy to the interface
 ip rsvp policy local identity RSVP-VIDEO
  maximum bandwidth group 12500
  forward all
  ! Defines a local RSVP policy to admit up to 12.5 Mbps of video flows
 ip rsvp policy local identity RSVP-VOICE
  maximum bandwidth group 2500
  forward all
  ! Defines a local RSVP policy to admit up to 2.5 Mbps of voice flows
 ip rsvp bandwidth 15000
  ! Specifies the amount of reservable BW (should match LLQ BW)
 ip rsvp signalling dscp 24
  ! Marks RSVP signaling traffic to CS3
 ip rsvp data-packet classification none
  ! Enables the IntServ/DiffServ model by disabling RSVP for classification
 ip rsvp resource-provider none
  ! Enables the IntServ/DiffServ model by disabling RSVP for scheduling
!
<snip>


! This section defines the regular expressions to match RSVP Application IDs
ip rsvp policy identity RSVP-VIDEO policy-locator .*VideoStream.*
 ! RSVP AppIDs with the string "VideoStream" will be
 ! Associated with the RSVP-VIDEO local RSVP policy
```

```
ip rsvp policy identity RSVP-VOICE policy-locator .*AudioStream.*
 ! RSVP AppIDs with the string "AudioStream" will be
 ! Associated with the RSVP-VIDEO local RSVP policy
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**
- **show interface**
- **show policy-map interface**
- **show ip rsvp interface**
- **show ip rsvp interface detail**
- **show ip rsvp installed**
- **show ip rsvp policy local** (as shown in Example 3-30)
- **show ip rsvp policy local detail** (as shown in Example 3-31)

Example 3-30 Verifying an Advanced IntServ/DiffServ RSVP Model with Local Policy and Application ID—show ip rsvp policy local

```
Router# show ip rsvp policy local
    A=Accept    F=Forward

    Serial2/0:
      Path:AF Resv:AF PathErr:AF ResvErr:AF ID(s): RSVP-VIDEO
      Path:AF Resv:AF PathErr:AF ResvErr:AF ID(s): RSVP-VOICE

Generic policy settings:
    Default policy: Accept all
    Preemption:     Disabled

Router#
```

Example 3-30 verifies that the local RSVP policy will accept and forward flows on Serial 2/0 that match the Application IDs defined in the RSVP-VIDEO and RSVP-VOICE policy identities. Additionally it confirms that the default policy is to accept all other RSVP requests and that preemption is disabled.

Example 3-31 Verifying an Advanced IntServ/DiffServ RSVP Model with Local Policy and Application ID—show ip rsvp policy local detail

```
Router# show ip rsvp policy local detail
Serial2/0:
  Policy for ID(s): RSVP-VIDEO

    Preemption Scope: Unrestricted.
    Local Override:   Disabled.
    Fast ReRoute:     Accept.
    Handle:           01000406.

                          Accept              Forward
    Path:                 Yes                 Yes
    Resv:                 Yes                 Yes
    PathError:            Yes                 Yes
    ResvError:            Yes                 Yes

                          Setup Priority      Hold Priority
    TE:                   N/A                 N/A
```

```
    Non-TE:                 N/A                   N/A

                            Current               Limit
    Senders:                12                  N/A
    Receivers:              12                  N/A
    Conversations:          12                  N/A
    Group bandwidth (bps):  9036K               12500K
    Per-flow b/w (bps):     753K                  N/A


  Policy for ID(s): RSVP-VOICE

    Preemption Scope: Unrestricted.
    Local Override:   Disabled.
    Fast ReRoute:     Accept.
    Handle:           02000403.

                            Accept              Forward
    Path:                   Yes                 Yes
    Resv:                   Yes                 Yes
    PathError:              Yes                 Yes
    ResvError:              Yes                 Yes

                            Setup Priority      Hold Priority
    TE:                     N/A                 N/A
    Non-TE:                 N/A                 N/A

                            Current               Limit
    Senders:                12                  N/A
    Receivers:              12                   N/A
    Conversations:          12                    N/A
    Group bandwidth (bps):  960K                2500K
    Per-flow b/w (bps):     80K                   N/A


Generic policy settings:
    Default policy: Accept all
    Preemption:     Disabled

Router#
```

Example 3-31 provides additional detail into each local policy highlighting that there are 12 video flows (at 753 kbps each) and 12 voice flows (at 80 kbps each) that are currently allocated to the video and voice pools, respectively.

Control Plane Policing Design

There are two main steps to deploying IOS Control Plane Policing policies:

- Defining CPP Traffic Classes
- Deploying CPP Policies

Each of these steps is detailed in the following sections.

Defining CPP Traffic Classes

Developing a CPP policy starts with the classification of the control plane traffic. To that end, the control plane traffic needs to be first identified and separated into different class maps.

This section presents a classification template that can be used as a model when implementing CPP on IOS routers in a compatible manner as was presented in Chapter 2, "Medianet Campus QoS Design 4.0," for Cisco Catalyst switches. This template presents a realistic classification, where traffic is grouped based on its relative importance and protocol type. The template uses eight different classes, which provide a high-level of granularity and make it suitable for real-world environments. It is important to note that, even though you can use this template as a reference, the actual number and type of classes needed for a given network can differ and should be selected based on local requirements, security policies, and a thorough analysis of baseline traffic.

This CPP template defines these eight traffic classes:

- Border Gateway Protocol (BGP)—This class defines traffic that is crucial to maintaining neighbor relationships for BGP routing protocol, such as BGP keepalives and routing updates. Maintaining BGP routing protocol is crucial to maintaining connectivity within a network or to an ISP. Sites that are not running BGP would not use this class.
- Interior Gateway Protocol (IGP)—This class defines traffic that is crucial to maintaining IGP routing protocols, such as Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and Routing Information Protocol (RIP). Maintaining IGP routing protocols is crucial to maintaining connectivity within a network.
- Interactive Management—This class defines interactive traffic that is required for day-to-day network operations. This class would include light volume traffic used for remote network access and management. For example, telnet, Secure Shell (SSH), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), and Terminal Access Controller Access Control System (TACACS).
- File Management—This class defines high volume traffic used for software image and configuration maintenance. This class would include traffic generated for remote file transfer, for example Trivial File Transfer Protocol (TFTP) and File Transfer Protocol (FTP).
- Monitoring/Reporting—This class defines traffic used for monitoring a router. This kind of traffic should be permitted but should never be allowed to pose a risk to the router. With CPP, this traffic can be permitted but limited to a low rate. Examples would include packets generated by ICMP echo requests (ping and trace route) as well as include traffic generated by Cisco IOS IP Service Level Agreements (IP SLAs) to generate ICMP with different DSCP settings in order to report on response times within different QoS data classes.
- Critical Applications—This class defines application traffic that is crucial to a specific network. The protocols that might be included in this class include generic routing encapsulation (GRE), Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), Gateway Load Balancing Protocol (GLBP), Session Initiation Protocol (SIP), Data Link Switching (DLSw), Dynamic Host Configuration Protocol (DHCP), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), Protocol Independent Multicast (PIM), multicast Traffic, and IPSec.
- Undesirable—This explicitly identifies unwanted or malicious traffic that should be dropped and denied access to the RP. For example, this class could contain packets from a well-known worm. This class is particularly useful when specific traffic destined to the router should always be denied rather than be placed into a default category. Explicitly denying traffic allows you to collect rough statistics on this traffic using **show** commands and thereby offers some insight into the rate of denied traffic. Access-list Entries (ACEs) used for classifying undesirable traffic may be added and modified as new undesirable applications appear on the network and as such these ACEs can be used as a reaction tool.
- Default—This class defines all remaining traffic destined to the route processor (RP) that does not match any other class. MQC provides the default class so you can specify how to treat traffic that is not explicitly associated with any other user-defined classes. It is desirable to give such traffic access

to the RP, but at a highly reduced rate. With a default classification in place, statistics can be monitored to determine the rate of otherwise unidentified traffic destined to the control plane. After this traffic is identified, further analysis can be performed to classify it. If needed, the other CPP policy entries can be updated to account for this traffic.

Deploying CPP Policies

Because CPP filters traffic, it is critical to gain an adequate level of understanding about the legitimate traffic destined to the RP prior to deployment. CPP policies built without proper understanding of the protocols, devices, or required traffic rates involved can block critical traffic, which has the potential of creating a DoS condition. Determining the exact traffic profile needed to build the CPP policies might be difficult in some networks.

The following steps employ a conservative methodology that facilitates the process of designing and deploying CPP. This methodology uses iterative ACL configurations to help identify and to incrementally filter traffic.

To deploy CPP, it is recommended that you perform these steps:

Step 1 Determine the classification scheme for your network.

Identify the known protocols that access the RP and divide them into categories using the most useful criteria for your specific network. As an example of classification, the nine categories template presented earlier in this section (BGP, IGP, interactive management, file management, reporting, critical applications, undesirable, and default) use a combination of relative importance and traffic type. Select a scheme suited to your specific network, which might require a larger or smaller number of classes.

Step 2 Define classification access lists.

Configure each ACL to permit all known protocols in its class that require access to the RP. At this point, each ACL entry should have both source and destination addresses set to any. In addition, the ACL for the default class should be configured with a single entry, **permit ip any any**. This matches traffic not explicitly permitted by entries in the other ACLs. After the ACLs have been configured, create a class map for each class defined in Step 1, including one for the default class. Then assign each ACL to its corresponding class map.

Note In this step you should create a separate class map for the default class, rather than using the class default available in some platforms. Creating a separate class map and assigning a **permit ip any any** ACL allows you to identify traffic not yet classified as part of another class.

Each class map should then be associated with a policy map that permits all traffic, regardless of classification. The policy for each class should be set as conform-action transmit exceed-action transmit.

Step 3 Review the identified traffic and adjust the classification.

Ideally, the classification performed in Step 1 identified all required traffic destined to the router. However, realistically, not all required traffic is identified prior to deployment and the **permit ip any any** entry in the default class ACL logs a number of packet matches. Some form of analysis is required to determine the exact nature of the unclassified packets. For example, you can use the **show access-lists** command to see the entries in the ACLs that are in use and to identify any additional traffic sent to the RP. However, to analyze the unclassified traffic you can use one of these techniques:

- General ACL classification as described in Characterizing and Tracing Packet Floods Using Cisco Routers, which is available at: http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080149ad6.shtml.

- Packet analyzers

When traffic has been properly identified, adjust the class configuration accordingly. Remove the ACL entries for those protocols that are not used. Add a **permit any any** entry for each protocol just identified.

Step 4 Restrict a macro range of source addresses.

Refine the classification ACLs by only allowing the full range of the allocated CIDR block to be permitted as the source address. For example, if the network has been allocated 172.68.0.0/16, then permit source addresses from 172.68.0.0/16 where applicable.

This step provides data points for devices or users from outside the CIDR block that might be accessing the equipment. An external BGP (eBGP) peer requires an exception because the permitted source addresses for the session lies outside the CIDR block. This phase might be left on for a few days to collect data for the next phase of narrowing the ACL entries.

Step 5 Narrow the ACL permit statements to authorized source addresses.

Increasingly limit the source address in the classification ACLs to only permit sources that communicate with the RP. For example, only known network management stations should be permitted to access the SNMP ports on a router.

Step 6 Refine CPP policies by implementing rate limiting.

Use the **show policy-map control-plane** command to collect data about the actual policies in place. Analyze the packet count and rate information and develop a rate limiting policy accordingly. At this point, you might decide to remove the class map and ACL used for the classification of default traffic. If so, you should also replace the previously defined policy for the default class by the class default policy.

A tested and validated set of CPP rates are presented in Table 3-6. It is important to note that the values presented here are solely for illustration purposes, as every environment has different baselines.

Note Unlike Catalyst switch implementations of Control Plane Policing (which allow for policing rates to only be defined in bits-per-second), IOS CPP allows the ability to configure rate limits based on either bits-per-second (bps) or packets-per-second (pps). Rate-limiting utilizing a pps rate is preferred, since it is the per-packet processing that more significantly impacts CPU utilization than the bps rate.

Table 3-6 Example Control Plane Policing Rate Limits and Actions

Traffic Class	Rate (pps)	Rate (bps)	Conform Action	Exceed Action
Border Gateway Protocol	500	4,000,000	Transmit	Drop
Interior Gateway Protocol	50	300,000	Transmit	Drop
Interactive Management	100	500,000	Transmit	Drop
File management	500	6,000,000	Transmit	Drop
Monitoring	125	900,000	Transmit	Drop
Critical applications	125	900,000	Transmit	Drop
Undesirable	10[1]	32 kbps[1]	Drop	Drop
Default	100	500,000	Transmit	Drop

1. The policing rate for the Undesirable CPP class is effectively 0—regardless of whether pps or bps rates are used and also regardless of what values these rates are set to—since both the conforming and exceeding actions for this class are set to drop. However, the policer still performs a metering operation of the rates of these flows which is often useful for management purposes.

This CPP deployment model-with the aforementioned CPP traffic classes and corresponding rate limit-is used in the 8-Class CPP model illustrated in Example 3-32. It is recommended to include a "CPP" prefix in the ACL and class-map names to prevent any potential classification errors for similarly-named ACLs and class-maps used in data-plane policies.

Note In Example 3-32, the CPP policy is applied to the control plane interface in both the input direction and also the output direction. This is done to simplify the policy construct. Alternatively, advanced administrators may choose to define separate policing rates for the input and output Control Plane Policers.

Example 3-32 Cisco IOS (8-Class) CPP Configuration Example

```
 ! This section defines the CPP IP extended named access-Lists
Router(config)#ip access-list extended CPP-ACL-BGP
Router(config-ext-nacl)# remark BGP
Router(config-ext-nacl)# permit tcp host 192.168.1.1 host 10.1.1.1 eq bgp
Router(config-ext-nacl)# permit tcp host 192.168.1.1 eq bgp host 10.1.1.1

Router(config)#ip access-list extended CPP-ACL-IGP
Router(config-ext-nacl)# remark IGP (OSPF)
Router(config-ext-nacl)# permit ospf any host 224.0.0.5
Router(config-ext-nacl)# permit ospf any host 224.0.0.6
Router(config-ext-nacl)# permit ospf any any

Router(config)#ip access-list extended CPP-ACL-INTERACTIVE-MANAGEMENT
Router(config-ext-nacl)# remark TACACS (return traffic)
Router(config-ext-nacl)# permit tcp host 10.2.1.1 host 10.1.1.1 established
Router(config-ext-nacl)# remark SSH
Router(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 host 10.1.1.1 eq 22
Router(config-ext-nacl)# remark SNMP
Router(config-ext-nacl)# permit udp host 10.2.2.2 host 10.1.1.1 eq snmp
Router(config-ext-nacl)# remark NTP
Router(config-ext-nacl)# permit udp host 10.2.2.3 host 10.1.1.1 eq ntp

Router(config)#ip access-list extended CPP-ACL-FILE-MANAGEMENT
Router(config-ext-nacl)# remark (initiated) FTP (active and passive)
Router(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 eq 21 host 10.1.1.1 gt 1023
established
Router(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 eq 20 host 10.1.1.1 gt 1023
Router(config-ext-nacl)# permit tcp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023
established
Router(config-ext-nacl)# remark (initiated) TFTP
Router(config-ext-nacl)# permit udp 10.2.1.0 0.0.0.255 gt 1023 host 10.1.1.1 gt 1023

Router(config)#ip access-list extended CPP-ACL-MONITORING
Router(config-ext-nacl)# remark PING-ECHO
Router(config-ext-nacl)# permit icmp any any echo
Router(config-ext-nacl)# remark PING-ECHO-REPLY
Router(config-ext-nacl)# permit icmp any any echo-reply
Router(config-ext-nacl)# remark TRACEROUTE
Router(config-ext-nacl)# permit icmp any any ttl-exceeded
Router(config-ext-nacl)# permit icmp any any port-unreachable

Router(config)#ip access-list extended CPP-ACL-CRITICAL-APPLICATIONS
```

```
Router(config-ext-nacl)# remark HSRP
Router(config-ext-nacl)# permit ip any host 224.0.0.2
Router(config-ext-nacl)# remark DHCP
Router(config-ext-nacl)#$ host 0.0.0.0 host 255.255.255.255 eq bootps
Router(config-ext-nacl)# permit udp host 10.2.2.8 eq bootps any eq bootps

Router(config)#ip access-list extended CPP-ACL-UNDESIRABLE
Router(config-ext-nacl)# remark UNDESIRABLE
Router(config-ext-nacl)# permit udp any any eq 1434


 ! This section defines the CPP policy class-maps
Router(config)# class-map match-all CPP-ACL-BGP
Router(config-cmap)# match access-group name CPP-ACL-BGP
 ! Associates the CPP-ACL-BGP access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-IGP
Router(config-cmap)#  match access-group name CPP-ACL-IGP
 ! Associates the CPP-ACL-IGP access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-INTERACTIVE-MANAGEMENT
Router(config-cmap)#  match access-group name CPP-ACL-INTERACTIVE-MANAGEMENT
 ! Associates the CPP-ACL-INTERACTIVE-MANAGEMENT access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-FILE-MANAGEMENT
Router(config-cmap)#  match access-group name CPP-ACL-FILE-MANAGEMENT
 ! Associates the CPP-ACL-FILE-MANAGEMENT access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-MONITORING
Router(config-cmap)#  match access-group name CPP-ACL-MONITORING
 ! Associates the CPP-ACL-MONITORING access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-CRITICAL-APPLICATIONS
Router(config-cmap)#  match access-group name CPP-ACL-CRITICAL-APPLICATIONS
 ! Associates the CPP-ACL-CRITICAL-APPLICATIONS access-list and class-map

Router(config-cmap)# class-map match-all CPP-ACL-UNDESIRABLE
Router(config-cmap)#  match access-group name CPP-ACL-UNDESIRABLE
 ! Associates the CPP-ACL-UNDESIRABLE access-list and class-map



 ! This section defines the CPP policy policy-map
Router(config)# policy-map CPP-POLICY
Router(config-pmap)# class CPP-ACL-BGP
Router(config-pmap-c)# police rate 500 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices BGP control-plane traffic to 500 pps
Router(config-pmap)# class CPP-ACL-IGP
Router(config-pmap-c)# police rate 50 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices IGP control-plane traffic to 50 pps
Router(config-pmap)# class CPP-ACL-INTERACTIVE-MANAGEMENT
Router(config-pmap-c)# police rate 100 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Management control-plane traffic to 100 pps
Router(config-pmap)# class CPP-ACL-FILE-MANAGEMENT
Router(config-pmap-c)# police rate 500 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices File-Mgmt control-plane traffic to 500 pps
```

```
Router(config-pmap)# class CPP-ACL-MONITORING
Router(config-pmap-c)# police rate 125 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Monitoring control-plane traffic to 125 pps
Router(config-pmap)# class CPP-ACL-CRITICAL-APPLICATIONS
Router(config-pmap-c)# police rate 125 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Critical Applications control-plane traffic to 125 pps
Router(config-pmap)# class CPP-ACL-UNDESIRABLE
Router(config-pmap-c)# police rate 10 pps
Router(config-pmap-c-police)# conform-action drop
Router(config-pmap-c-police)# exceed-action drop
 ! Polices Undesirable control-plane traffic to (effectively) 0 pps
 ! As both the conform and exceed action are set to drop
Router(config-pmap)# class class-default
Router(config-pmap-c)# police rate 100 pps
Router(config-pmap-c-police)# conform-action transmit
Router(config-pmap-c-police)# exceed-action drop
 ! Polices all other control-plane traffic to 100 pps



! This section applies the CPP policy to the control-plane in both directions
Router(config)# control-plane
Router(config-cp)# service-policy input CPP-POLICY
 ! Attaches the CPP-POLICY to the control plane in the input direction
Router(config-cp)# service-policy output CPP-POLICY
 ! Attaches the CPP-POLICY to the control plane in the output direction
```

This configuration can be verified with the commands:

- **show class-map**
- **show policy-map**
- **show policy-map control-plane { input | output }** (as shown in Example 3-33)

Example 3-33 Verifying a Control Plane Policy—show policy-map control-plane

```
Router# show policy-map control-plane input
 Control Plane

  Service-policy input: CPP-POLICY

    Class-map: CPP-ACL-BGP (match-all)
      6552 packets, 1323504 bytes
      5 minute offered rate 24000 bps, drop rate 0 bps
      Match: access-group name CPP-ACL-BGP
      police:
          rate 500 pps, burst 122 packets
        conformed 6552 packets; actions:
          transmit
        exceeded 0 packets; actions:
          drop
        conformed 15 pps, exceed 0 pps

    Class-map: CPP-ACL-IGP (match-all)
      2162 packets, 367540 bytes
      5 minute offered rate 6000 bps, drop rate 0 bps
      Match: access-group name CPP-ACL-IGP
      police:
          rate 50 pps, burst 12 packets
```

```
    conformed 2162 packets; actions:
      transmit
    exceeded 0 packets; actions:
      drop
    conformed 5 pps, exceed 0 pps

Class-map: CPP-ACL-INTERACTIVE-MANAGEMENT (match-all)
  21621 packets, 11978034 bytes
  5 minute offered rate 198000 bps, drop rate 0 bps
  Match: access-group name CPP-ACL-INTERACTIVE-MANAGEMENT
  police:
      rate 100 pps, burst 24 packets
    conformed 21621 packets; actions:
      transmit
    exceeded 0 packets; actions:
      drop
    conformed 50 pps, exceed 0 pps

Class-map: CPP-ACL-FILE-MANAGEMENT (match-all)
  33263 packets, 47965246 bytes
  5 minute offered rate 727000 bps, drop rate 0 bps
  Match: access-group name CPP-ACL-FILE-MANAGEMENT
  police:
      rate 500 pps, burst 122 packets
    conformed 33263 packets; actions:
      transmit
    exceeded 0 packets; actions:
      drop
    conformed 76 pps, exceed 0 pps

Class-map: CPP-ACL-MONITORING (match-all)
  6552 packets, 694512 bytes
  5 minute offered rate 12000 bps, drop rate 0 bps
  Match: access-group name CPP-ACL-MONITORING
  police:
      rate 125 pps, burst 30 packets
    conformed 6552 packets; actions:
      transmit
    exceeded 0 packets; actions:
      drop
    conformed 15 pps, exceed 0 pps

Class-map: CPP-ACL-CRITICAL-APPLICATIONS (match-all)
  36035 packets, 7279070 bytes
  5 minute offered rate 133000 bps, drop rate 0 bps
  Match: access-group name CPP-ACL-CRITICAL-APPLICATIONS
  police:
      rate 125 pps, burst 30 packets
    conformed 36035 packets; actions:
      transmit
    exceeded 0 packets; actions:
      drop
    conformed 83 pps, exceed 0 pps

Class-map: CPP-ACL-UNDESIRABLE (match-all)
  432 packets, 349920 bytes
  5 minute offered rate 6000 bps, drop rate 0 bps
  Match: access-group name CPP-ACL-UNDESIRABLE
  police:
      rate 10 pps, burst 2 packets
    conformed 432 packets; actions:
      drop
    exceeded 0 packets; actions:
      drop
```

```
        conformed 1 pps, exceed 0 pps

    Class-map: class-default (match-any)
      1402 packets, 92916 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
      police:
          rate 100 pps, burst 24 packets
        conformed 1403 packets; actions:
          transmit
        exceeded 0 packets; actions:
          drop
        conformed 3 pps, exceed 0 pps
Router#
```

As show in Example 3-33, there are matches and conforming packets for each CPP class, but there are no exceeding drops on any class. If there were any exceeding drops, this would be an indication that the network administrator may need to review the respective policing rates.

Also, for the sake of brevity and to minimize redundancy, only the input CPP policy's output is displayed because the **show policy-map control-plane** command was used with the **input** keyword; however, the output policy's statistics are relatively similar.

Cisco ASR Internal QoS Design

The Cisco ASR 1000 performs QoS operations in both hardware and software, making it a very scalable, efficient, yet flexible platform for QoS. Due to the hardware architecture of this platform, there may be certain circumstances where this platform becomes internally oversubscribed. However, there are mechanisms in place that allow for administrators to address such oversubscription scenarios. Therefore, the administrator not only has to define the WAN edge policies described in this design chapter, but may also have to tune the internal scheduling mechanisms on the ASR 1000 to align.

The following sections describe the internal QoS architecture of the ASR 1000, highlighting the potential oversubscription points and present platform-specific CLI that can tune internal scheduling policies to match medianet requirements.

ASR QoS Architecture

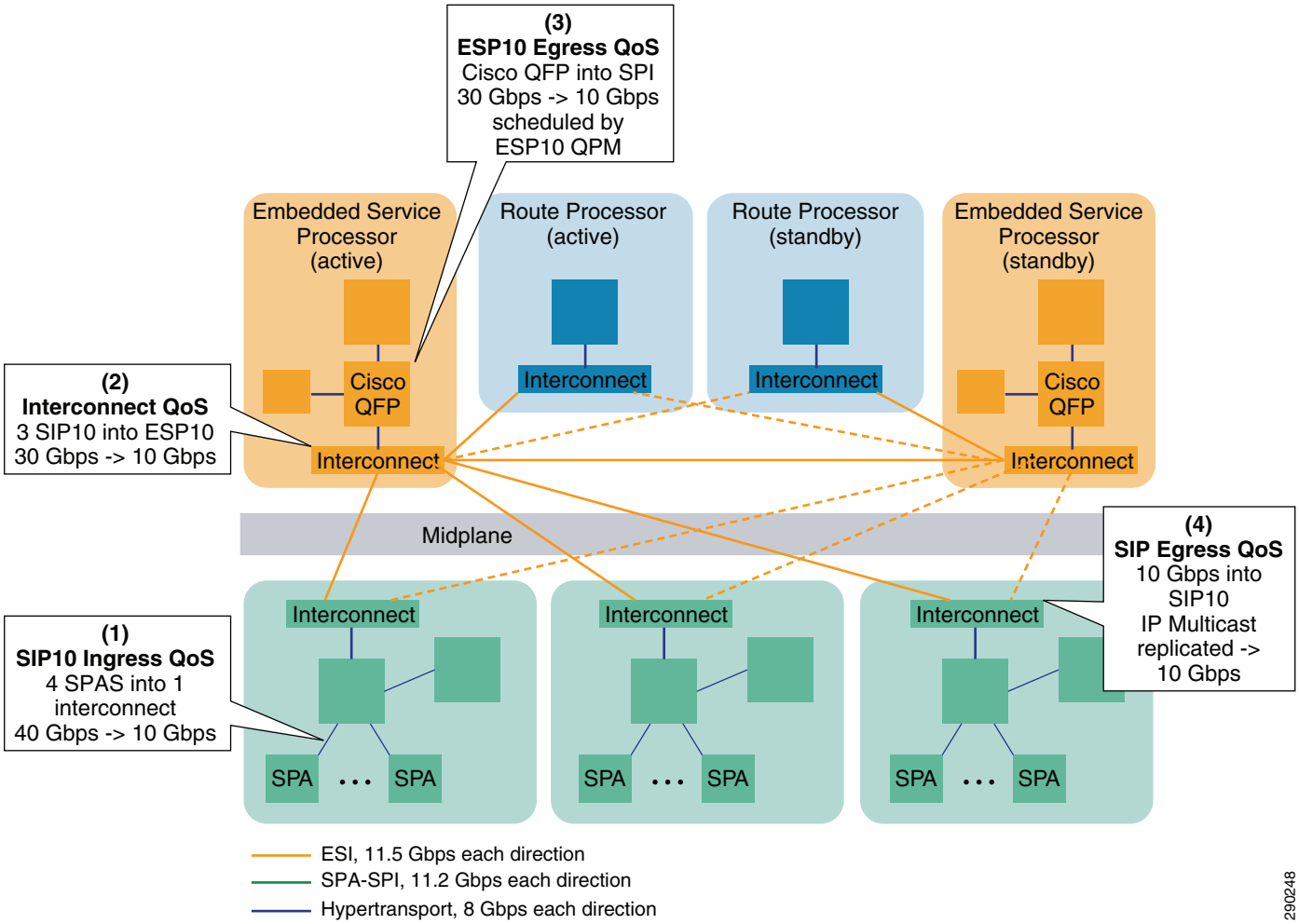
The Cisco ASR 1000 series routers utilize Embedded Services Processors (ESPs) to perform QoS operations. The main QoS component of ESPs is the Cisco QuantumFlow Processor (QFP) and its two chips: the Cisco QFP Engine and the Cisco QFP Traffic Manager. The Cisco QFP Engine is in charge of MQC classification, marking, WRED, policing; the Cisco QFP Traffic Manager handles shaping, priority and CBWFQ scheduling.

A significant advantage of the ASR 1000 architecture is the fact that Cisco QFP Traffic Manager is not affected by any of the features that Cisco QFP Engine needs to perform. So QoS shaping and Low Latency Queuing (LLQ) configurations has minimum impact performance in the Cisco ASR 1000. As such, while QoS performance on the ASR depends upon the QoS configuration and the combination of other enabled features, it is very scalable. For example, the performance on an ASR 1000 with ESP10 with a full 12-class medianet QoS configuration on several thousand subinterfaces, running concurrently with IPv6 multicast and access control lists is several million packets per second

Additionally, the Cisco ASR 1000 SIPs are also capable of ingress classification, buffering, and scheduling with a non-MQC configuration.

To better understand the QoS architecture of the ASR 1000 platform, consider the flow of packets from ingress to egress, as shown in Figure 3-22, which shows the packet flow through an ASR 1000 with ESP10 with three SIP-10 processors.

Figure 3-22 Cisco ASR 1000 Internal QoS Flow



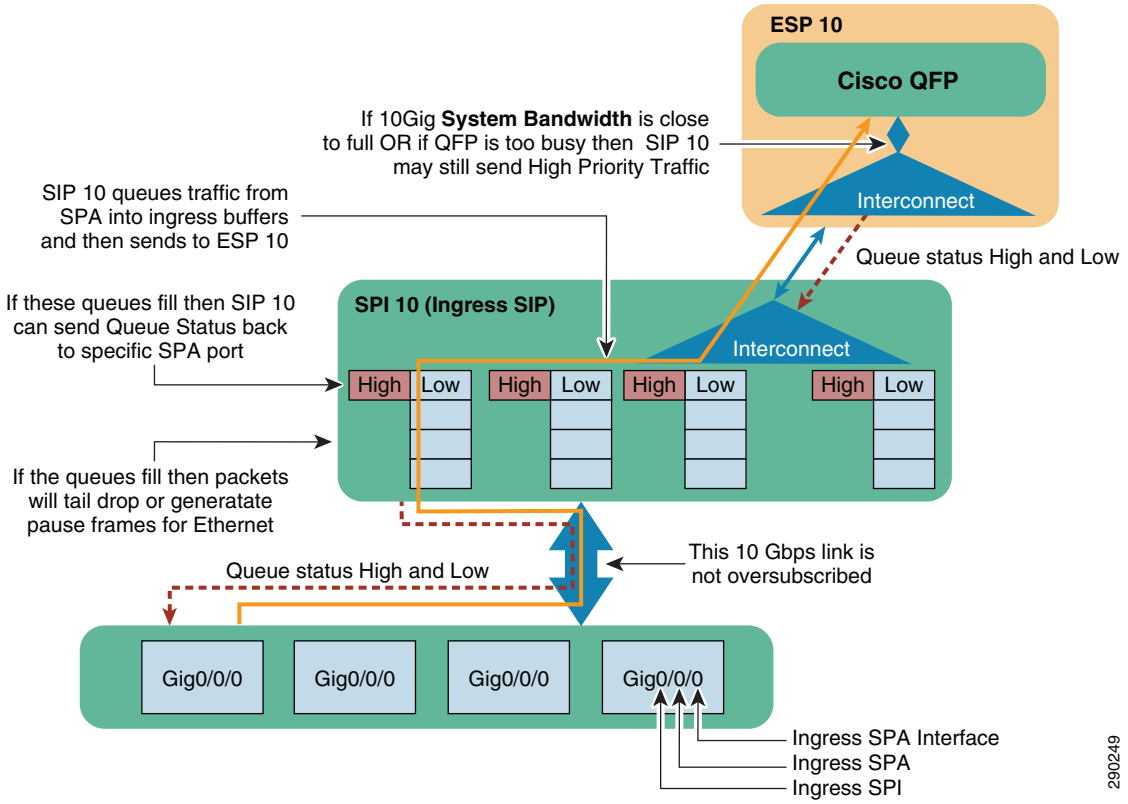
Each of the callout boxes in Figure 3-22 represents a possible internal congestion point internal to the ASR 1000 with an ESP 10. Specifically, these potential congestion points are:

1. SIP10 ingress classification and scheduling—This point represents a possible 4:1 oversubscription point (up to 40 Gbps of SPA interfaces contending for a 10 Gbps interconnect).
2. ESP10 interconnect scheduling—This point represents a possible 3:1 oversubscription point (3 SIP10s contending for a 10 Gbps interconnect).
3. ESP10 egress QoS—This point represents a possible 3:1 oversubscription point (30 Gbps from the QFP contending for a 10 Gbps interconnect).
4. SIP10 buffering and egress QoS—This point represents a possible 1:1+ oversubscription point (10 Gbps from the interconnect contending for 10 Gbps of SPA interfaces, which may include the requirement to IP replicate multicast packets).

To deal with these potential oversubscription scenarios, the Cisco ASR 1000 uses a two-queue internal bandwidth scheduler to prioritize packets: one queue operates as a strict priority queue, while the other operates as a low-priority queue.

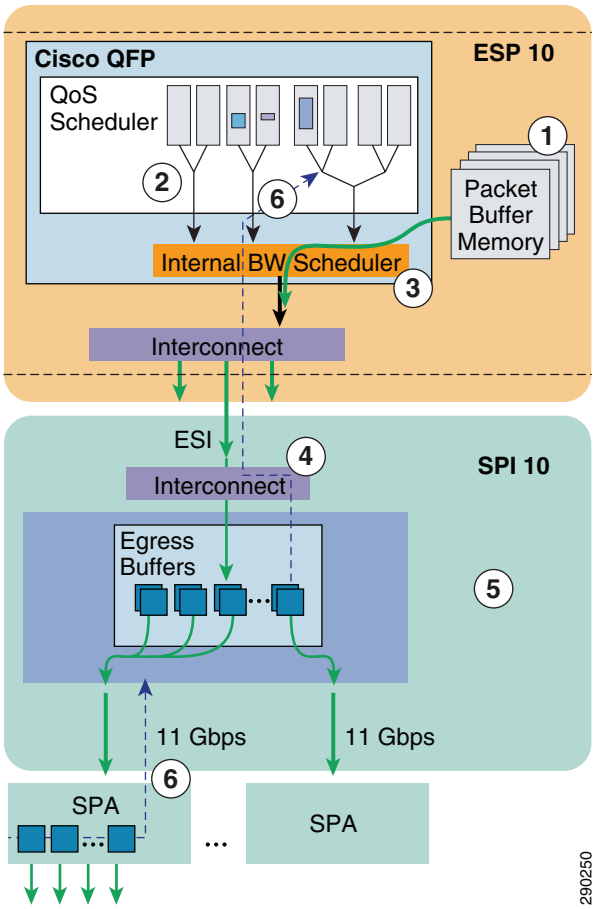
Figure 3-23 shows a basic overview of how this internal queuing system operates on ingress and highlights how the system is constantly aware of the queue status. Because of this internal queue status communication, high priority packets can always be processed-even when the interconnect is overcommitted-since the interconnect uses backpressure from the Cisco QFP to delay low priority packets so that high priority packets continue to flow.

Figure 3-23 Cisco ASR 1000 Internal Ingress Scheduling Operation



In the opposite direction, Figure 3-24 illustrates how scheduling takes place on egress. Within the Cisco QFP, the QFP Traffic Manager Scheduler performs packet-scheduling decisions based on MQC policies. Then, the internal bandwidth scheduler allocates the egress packets (from the QFP) among the SIPs, selecting first the internally-designated high priority packets; excess bandwidth-bandwidth remaining after the internally-designated high priority packets have been serviced-is evenly shared among the SIPs; that is, unless the administrator has tuned internal scheduling weights or set internal scheduling minimum bandwidth guarantees per SIP (which will be discussed later). Finally, shallow buffers on the SIP and SPAs are used to allow simultaneous packet transfer out of multiple ports

Figure 3-24 Cisco ASR 1000 Internal Egress Scheduling Operation



However, it should be noted that not every combination of SIPs and ESP poses a potential oversubscription scenario. Specifically, Table 3-7 details the oversubscription scenarios for SIP-10s and Table 3-8 details the oversubscription scenarios for SIP-40s.

Table 3-7 Cisco ASR 1000 SIP-10 Oversubscription Scenarios

ASR 1000 Chassis Type	ESP Type	Oversubscription State

Table 3-7 Cisco ASR 1000 SIP-10 Oversubscription Scenarios

ASR1000-SIP10G • Incoming rate from SPAs. Maximum 4 multiplied by 11.2 Gbps • Outgoing rate towards ESP 11.2 Gbps **Conclusion: SIP10G is always oversubscribed.**	ESP-2.5 G and ESP-5 G (Supported only on ASR-1002)	No ESP oversubscription. ESPs interconnect device handles the entire 10G incoming traffic.
	ESP-10G	**ESP is oversubscribed.** Example: Input rate from (example) 3 carrier cards is 3 multiplied by 11.2Gbps Output rate towards QFP=12.8Gbps
	ESP-20G	**ESP is oversubscribed.** Example: Input rate from (example) 3 carrier cards is 3 multiplied by 11.2Gbps Output rate towards QFP=25.6 Gbps
	ESP-40G	No ESP oversubscription. Example: Input rate from (example) 3 carrier cards is 3 multiplied by 11.2Gbps Output rate towards QFP=2*25.6 Gbps

Table 3-8 Cisco ASR 1000 SIP-40 Oversubscription Scenarios

ASR 1000 Chassis Type	ESP Type	Oversubscription State

Table 3-8 Cisco ASR 1000 SIP-40 Oversubscription Scenarios

ASR1000-SIP40G • Incoming rate from SPAs. Maximum 4 multiplied by 11.2 Gbps • Outgoing rate towards ESP depends on the ESP type **Conclusion: SIP40G oversubscription depends on the ESP type.**	ESP-2.5 G and ESP-5 G (Supported only on ASR-1002)	Not supported.
	ESP-10G	ASR1000-SIP40G operates as ASR1000-SIP10G. Example: Input rate from (example) 3 carrier cards is 3 multiplied by 11.2 Gbps Output rate towards QFP=12.8 Gbps **ESP is oversubscribed.**
	ESP-20G	ASR1000-SIP40G operates as ASR1000-SIP10G. Example: Input rate from (example) 3 carrier cards is 3 multiplied by 11.2 Gbps Output rate towards QFP=25.6 Gbps **ESP is oversubscribed.**
	ESP-40G	ASR1000-SIP40G operates as ASR1000-SIP40G. Example: Input rate from (example) 3 carrier cards is 3 multiplied by 23 Gbps Output rate towards QFP=51.2 Gbps ESP is oversubscribed.

Nonetheless, even though oversubscription scenarios are possible on the ASR 1000 platforms, there are internal schedulers that can prioritize application classes, which will be discussed next.

ASR Internal QoS Design

While the ASR 1000 internal QoS architecture may seem complex, the tasks required to configure internal hardware queuing inline with medianet QoS policies is relatively simple: the main requirement is to identify—at either the SPA- or the SIP-level—the packets that are to be assigned to the strict-priority internal queue.

Note Optionally, the scheduling of low-priority packets may also be weighted and/or guaranteed on a per SIP or SPA-basis. However, the design principles in this chapter are based on application-priority (as indicated by DSCP values) rather than by interface-priority. Therefore, this type of low-priority

scheduling tuning is not covered in this design chapter, but can be referenced in the documentation at: http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html#wp1292478.

Ethernet SPAs and ATM SPAs require internal priority classification at the SPA-level; on the other hand, POS SPAs, channelized SPAs and clear-channel SPAs require internal priority classification at the SIP-level. A full matrix of SPAs by internal classification schemes is shown in Table 3-9.

Table 3-9 SPA-Based Matrix of Ingress Classification by SIP- or SPA-Level

Classification at the SPA Level	Classification at the SIP Level
Ethernet SPAs • SPA-4X1FE-TX-V2 • SPA-8X1FE-TX-V2 • SPA-2X1GE-V2 • SPA-5X1GE-V2 • SPA-8X1GE-V2 • SPA-10X1GE-V2 • SPA-1X10GE-L-V2	Serial and Channelized SPA • SPA-2XCT3/DS0 • SPA-4XCT3/DS0 • SPA-8XCHT1/E1 • SPA-1XCHSTM1/OC3 • SPA-1XCHOC12/DS0 • SPA-2xT3/E3 • SPA-4xT3/E3 • SPA-4xT-SERIAL
ATM SPAs • SPA-1XOC3-ATM-V2 • SPA-3XOC3-ATM-V2 • SPA-1XOC12-ATM-V2	• POS SPAs • SPA-2XOC3-POS • SPA-4XOC3-POS • SPA-1XOC12-POS • SPA-2XOC12-POS • SPA-4XOC12-POS • SPA-8XOC12-POS • SPA-8XOC3-POS • SPA-1XOC48POS/RPR • SPA-2XOC48POS/RPR • SPA-4XOC48POS/RPR • SPA-OC192POS-XFP

Both SPA-Level and SIP-Level internal scheduling classification will now be detailed.

SPA-Based ASR Internal Scheduling Classification

Ethernet and ATM SPAs perform classification at the SPA level. In the SPA-based classification model, the SPA performs both Layer 2 and Layer 3 classification, and decides on the internal priority of the packet. After classifying the packets into high priority and low priority, the SPA has unique channels per priority: all high-priority packets are sent on separate channels than low-priority packets. In such a

scenario, the SPA queues the packets on high channels to high-priority buffers and low-channels to low-priority buffers. After the packets are classified into high priority and low priority, the packets are sent to the ESP for further processing.

Internal classification can be based on DSCP, IPv6 traffic class, MPLS EXP or 802.1Q/p CoS values. Consistent with the designs presented in this chapter, DSCP-based classification will be used in these examples.

Note For classification based on IPv6 traffic classes or MPLS EXP bits or by 802.1Q/p CoS values, refer to the feature documentation at: http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html.

The ASR-specific CLI for enabling SPA-based internal scheduling classification is the use of the **plim qos input map** interface-configuration command, as shown in Example 3-34. By default EF is mapped to the strict priority internal queue and all other DSCP values are mapped to the low-priority internal queue. Therefore, the only amendment needed from these default settings is to map CS4 (Realtime Interactive) and CS5 (Broadcast Video) to the strict-priority internal queue.

Example 3-34 ASR 1000 SPA-Based Internal Scheduling Classification Example

```
ASR(config)# interface GigabitEthernet0/0/0
ASR(config-if)# plim qos input map ip dscp-based
 ! Designates that internal scheduling is to be DSCP-based
ASR(config-if)# plim qos input map ip dscp  cs4 cs5 ef queue strict-priority
 ! Maps CS4 (Realtime Interactive) & CS5 (Broadcast Video) to the internal PQ
 ! EF is already mapped to the internal PQ by default
```

Note It should be noted CS4 and CS5 will be converted to 32 and 40 (respectively) in the ASR configuration. Additionally, neither EF (nor 46) will appear in the configuration, as DSCP EF/46 is mapped to the internal priority queue by default. Including it in Example 3-34 has been done simply for the sake of consistency with previous examples that map all three values to the strict-priority queue.

This configuration can be verified with the command:

- **show platform hardware interface** *type sip/spa/interface* **plim qos input map** (as shown in Example 3-35)

Example 3-35 Verifying ASR SPA-Based Ingress Scheduling Classification—show platform hardware interface plim qos input map

```
ASR# show platform hardware interface Gig 0/0/0 plim qos input map
Interface GigabitEthernet0/0/0
    Low Latency Queue(High Priority):
    IP DSCP, 32, 40, 46
    IPv6 TC, 46
    MPLS EXP, 6, 7

ASR#
```

As shown in Example 3-35, DSCP values 32 (CS4), 40 (CS5), and EF (46) will be mapped to the internal priority queue on this ASR.

SIP-Based ASR Internal Scheduling Classification

POS SPAs, channelized, and clear-channel SPAs support packet classification at SIP level. In SIP-based classification, the SIP does the classification for SPAs and classifies the packet into high-priority and low-priority.

Commands have been introduced in IOS XE Release 3.1S to configure SIP based classification. To classify high-priority packets a classification template is defined using the **ingress-class map** index command. The classification template-specific details are defined inside the template, and the template is attached to an interface using the **plim qos class-map** index interface configuration command.

Similar to the SPA-based internal scheduling classification example, DSCP-based classification is used in this SIP-based example. By default, EF is mapped to the strict priority internal queue and all other DSCP values are mapped to the low-priority internal queue. Therefore, the only amendment needed from these default settings is to map CS4 (Realtime Interactive) and CS5 (Broadcast Video) to the strict-priority internal queue via the **ingress-class-map** index, as shown in Example 3-36.

Example 3-36 ASR 1000 SIP-Based Internal Scheduling Classification Example

```
ASR(config)# ingress-class-map 1
ASR(config-ing-class-map)# map ip dscp-based
ASR(config-ing-class-map)# map ip dscp 32 40 queue strict-priority
…
ASR(config)# interface POS0/1/0
ASR(config-if)# plim qos input class-map 1
```

This configuration can be verified with the command:

- **show platform hardware interface** *type sip/spa/interface* **plim qos input map** (as shown in Example 3-37)

Example 3-37 Verifying ASR SIP-Based Ingress Scheduling Classification—show platform hardware interface plim qos input map

```
ASR# show platform hardware interface pos 0/1/0 plim qos input map
Interface POS0/1/0 uses
Ingress-class-map 1
High-priority queue settings:
  IPv4 DSCP: 32 40 46
  IPv6 TC: 46
  MPLS EXP: 6 7
ASR#
```

As shown in Example 3-37, interface POS 0/1/0 references the **ingress-class-map 1** index (associated with its SIP) which maps DSCP values 32 (CS4), 40 (CS5) and EF (46) to the internal priority queue on this ASR.

Summary

Wide area network edges usually represent the greatest bottlenecks in an enterprise network and therefore require the most attention to QoS design. Two strategic QoS design principles that apply to WAN edges are to enable queuing policies are every node where the potential for congestion exists, and to protect the control plane and data plane.

This design chapter began with an overview of QoS design considerations specific to the wide-area network which apply to both Layer 2 private WANs and Layer 3 VPNs, with the clearly-defined focus and scope of this chapter being the former. Routing platforms for medianet WANs were identified, including the Cisco 7200VXR NPE-G2 routers, the 6500/7600 routers with SIPs and WAN SPAs, and the Cisco ASR 1000 routers with ESPs. The design implications of hardware vs. software QoS were considered, as well as platform-specific performance guidelines were shared. Following this, service level attributes, such as latency and jitter, were considered as to how these impact WAN QoS designs.

Subsequently WAN-specific QoS tools were reviewed, including the Tx-Ring, CBWFQ, LLQ, WRED, PAK_Priority and RSVP. The inner workings of these mechanisms were discussed along with their design implications. These considerations led to the definition of the QoS roles of WAN edges, which require that:

- Egress LLQ/CBWFQ policies **should** be applied on these interfaces, as these LAN/WAN edges will most often represent a speed-mismatch and thus a congestion/queuing scenario.
- RSVP policies **may** be enabled on these interfaces to provide network-aware dynamic admission control functionality. RSVP policies **may** include Application ID RSVP policies.

Following this, WAN edge QoS design principles were reviewed. These design principles include:

- Limiting the amount of strict priority queuing to 33% of link bandwidth capacity
- Provisioning at least 25% of a link's bandwidth capacity for best effort traffic
- Assigning a minimum amount of bandwidth—such as 1%—to the Scavenger class (if enabled)
- Enable fair-queuing pre-sorters on both all CBWFQ traffic classes, except for the control and scavenger classes
- Enable DSCP-based WRED on all AF traffic classes
- Consider the ongoing evolution of business requirements and how these may affect future WAN edge class migration models

With these principles in mind, three Differentiated Services WAN edge models were presented in detail, including a 4-class model, an 8-class model and a 12-class model. These models were complemented with verification command outputs to ensure that the policies were working as intended. Additionally, some caveats and policy-amendments to the SIP/SPA implementations of these designs were presented, as were link-specific aspects of applying these policies to serial, ATM and POS links.

The following section shifted gears from DiffServ to IntServ, highlighting the need for dynamic network-aware admission control to support rich-media applications. Two RSVP models were detailed: a basic IntServ/DiffServ model and a more advanced IntServ/DiffServ model with Application IDs. Furthermore, it was noted that RSVP will likely pay an increasingly significant role in future medianet designs.

Next, consideration and design recommendations was given for hardening the wide area network via control-plane policing policies, which included defining an 8-class CPP model complete with recommended initial policing rates, as well as deployment recommendations for applying and fine-tuning these CPP policies.

Finally, the internal QoS architecture of the ASR 1000 platforms with ESPs was overviewed to draw attention to the need for internal QoS on these potentially-oversubscribed platforms. Two ASR internal QoS model were detailed: a SPA-based model and a SIP-based model.

References

IETF RFCs

- RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
  http://www.ietf.org/rfc/rfc2474
- RFC 2597 Assured Forwarding PHB Group
  http://www.ietf.org/rfc/rfc2597
- RFC 2615 PPP over SONET/SDH
  http://www.ietf.org/rfc/rfc2615
- RFC 2697 A Single Rate Three Color Marker
  http://www.ietf.org/rfc/rfc2697
- RFC 2698 A Two Rate Three Color Marker
  http://www.ietf.org/rfc/rfc2698
- RFC 2872 Application and Sub Application Identity Policy Element for Use with RSVP
  http://www.ietf.org/rfc/rfc2872
- RFC 3168 The Addition of Explicit Congestion Notification (ECN) to IP
  http://www.ietf.org/rfc/rfc3168
- RFC 3246 An Expedited Forwarding PHB (Per-Hop Behavior)
  http://www.ietf.org/rfc/rfc3246
- RFC 3662 A Lower Effort Per-Domain Behavior for Differentiated Services
  http://www.ietf.org/rfc/rfc36
- RFC 4594 Configuration Guidelines for DiffServ Service Classes
  http://www.ietf.org/rfc/rfc4594

Cisco Documentation

- Cisco IOS Quality of Service Solutions Configuration Guide, Release 15.0
  http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/15_0/qos_15_0_book.html
- Cisco IOS Hardware and Software Interface Command Reference: TX-Ring Limit
  http://www.cisco.com/en/US/docs/ios/interface/command/reference/ir_t2.html#wp1013139
- Cisco 6500 Documentation: Configuring QoS Features on a SIP
  http://www.cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/6500series/76cfgsip.html#wp1177665
- Cisco 7600 Documentation: Configuring QoS Features on a SIP
  http://www.cisco.com/univercd/cc/td/doc/product/core/cis7600/76sipspa/sipspasw/76sipssc/76cfgsip.htm#wp1162382
- Cisco ASR 1000 Series Aggregation Services Routers SIP and SPA Software Configuration Guide: Classifying and Scheduling Packets for ASR 1000 Series
  http://www.cisco.com/en/US/partner/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRimpqos.html

White Papers

- Cisco Unified Communications System 8.x SRND: Call Admission Control Chapter
  http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/srnd/8x/cac.html
- Deploying Control Plane Policing
  http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6586/ps6642/prod_white_paper0900aecd804fa16a.html
- Cisco ASR 1000 Series Aggregation Services Routers: QoS Architecture and Solutions
  http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-449961_ps9343_Product_Solution_Overview.html