

Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which packets are transmitted out an interface based on priorities assigned to those packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the packet's classification, and scheduling of the packets in a queue for transmission. The congestion management QoS feature offers four types of queueing protocols, each of which allows you to specify creation of a different number of queues, affording greater or lesser degrees of differentiation of traffic and the order in which that traffic is transmitted.

During periods with light traffic, that is, when no congestion exists, packets are transmitted out the interface as soon as they arrive. During periods of transmit congestion at the outgoing interface, packets arrive faster than the interface can transmit them. If you use congestion management features, packets accumulating at an interface are queued until the interface is free to transmit them; they are then scheduled for transmission according to their assigned priority and the queueing mechanism configured for the interface. The router determines the order of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

This chapter discusses these four types of queueing, which constitute the congestion management QoS feature:

- **First-In, First-Out Queueing (FIFO).** FIFO entails no concept of priority or classes of traffic. With FIFO, transmission of packets out the interface occurs in the order the packets arrive.
- **Weighted Fair Queueing (WFQ).** WFQ offers dynamic, fair queueing that divides bandwidth across queues of traffic based on weights. WFQ ensures that all traffic is treated fairly, given its weight. To help understand how WFQ works, consider the queue for a train of File Transfer Protocol (FTP) packets as a queue for the collective and the queue for discrete interactive traffic packets as a queue for the individual. Given the weight of the queues, WFQ ensures that for all FTP packets transmitted as a collective an equal number of individual interactive traffic packets are transmitted.

Given this handling, WFQ ensures satisfactory response time to critical applications, such as interactive, transaction-based applications, that are intolerant of performance degradation. For serial interfaces at E1 (2.048 Mbps) and below, WFQ is used by default. When no other queueing strategies are configured, all other interfaces use FIFO by default.

- **Custom Queueing (CQ).** With CQ, bandwidth is allocated proportionally for each different class of traffic. CQ allows you to specify the number of bytes or packets to be drawn from the queue, which is especially useful on slow interfaces.
- **Priority Queueing (PQ).** With PQ, packets belonging to one priority class of traffic are transmitted before all lower priority traffic to ensure timely delivery of those packets.

Note You can assign only one queueing mechanism type to an interface.

Note A variety of queueing mechanisms can be configured using multilink, for example, Multichassis Multilink PPP (MMP). However, if only PPP is used on a tunneled interface—for example, virtual private dialup network (VPND), PPP over Ethernet (PPPoE), or PPP over Frame Relay (PPPoFR)—no queueing can be configured on the virtual interface.

Why Use Congestion Management?

Today's heterogeneous networks include many different protocols used by applications, giving rise to the need to prioritize traffic in order to satisfy time-critical applications while still addressing the needs of less time-dependent applications, such as file transfer. Different types of traffic sharing a data path through the network can interact with one another in ways that affect their application performance. If your network is designed to support different traffic types that share a single data path between routers, you should consider using congestion management techniques to ensure fairness of treatment across the various traffic types.

Here are some broad ideas to consider in determining if you need to configure congestion management QoS:

- Traffic prioritization is especially important for delay-sensitive, interactive transaction-based applications—for instance, desktop video conferencing—that require higher priority than do file transfer applications. However, use of WFQ ensures that all traffic is treated fairly, given its weight, and in a dynamic manner. For example, WFQ addresses the requirements of the interactive application without penalizing the FTP application.
- Prioritization is most effective on WAN links where the combination of bursty traffic and relatively lower data rates can cause temporary congestion.
- Depending on the average packet size, prioritization is most effective when applied to links at T1/E1 bandwidth speeds or lower.
- If users of applications running across your network notice poor response time, you should consider using congestion management features. Congestion management features are dynamic, tailoring themselves to the existing network conditions. However, consider that if a WAN link is constantly congested, traffic prioritization may *not* resolve the problem. Adding bandwidth might be the appropriate solution.
- If there is no congestion on the WAN link, there is no reason to implement traffic prioritization.

Here are some steps that summarize aspects you should consider in determining whether you should establish and implement a queueing policy for your network:

- Step 1** Determine if the WAN is congested—that is, whether users of certain application perceive a performance degradation.
- Step 2** Determine your goals and objectives based on the mix of traffic you need to manage and your network topology and design. In identifying what you want to achieve, consider whether your goal is among the following:
- To establish fair distribution of bandwidth allocation across all of the types of traffic you identify.

- To grant strict priority to traffic from special kinds of applications you service—for example, interactive multimedia applications—possibly at the expense of less critical traffic you also support.
- To customize bandwidth allocation so that network resources are shared among all of the applications you service, each having the specific bandwidth requirements you have identified.

To effectively configure queueing, you must analyze the types of traffic using the interface and determine how to distinguish them. See the chapter “Classification Overview” for a description of how packets are classified.

After you assess your needs, review the available congestion management queueing mechanisms described in this chapter and determine which approach best addresses your requirements and goals.

Step 3 Configure the interface for the kind of queueing strategy you have chosen, and observe the results.

Traffic patterns change over time, so you should repeat the analysis process described in Step 2 periodically, and adapt the queueing configuration accordingly.

See the section “Deciding Which Queueing Policy to Use” for elaboration of the differences between the various queueing mechanisms.

Deciding Which Queueing Policy to Use

This section looks briefly at some of the differences between the types of queueing and includes a table that compares the three main queueing strategies.

FIFO queueing performs no prioritization of data packets on user data traffic. It entails no concept of priority or classes of traffic. When FIFO is used, ill-behaved sources can consume available bandwidth, bursty sources can cause delays in time-sensitive or important traffic, and important traffic may be dropped because less important traffic fills the queue.

Consider these differences in deciding whether to use CQ or PQ:

- CQ guarantees some level of service to all traffic because you can allocate bandwidth to all classes of traffic. You can define the size of the queue by determining its configured packet-count capacity, thereby controlling bandwidth access.
- PQ guarantees strict priority in that it ensures that one type of traffic will be transmitted, possibly at the expense of all others. For PQ, a low priority queue can be detrimentally affected, and, in the worst case, never allowed to transmit its packets if there is a limited amount of available bandwidth or if the transmission frequency of critical traffic is high.

In deciding whether to use WFQ or one of the other two queueing types, consider these differences in WFQ and PQ and CQ:

- WFQ does not require configuration of access lists to determine the preferred traffic on a serial interface. Rather, the fair queue algorithm dynamically sorts traffic into messages that are part of a conversation.
- Low-volume, interactive traffic gets fair allocation of bandwidth with WFQ, as does high-volume traffic such as file transfers.

Table 5 compares the salient features of WFQ, CQ, and PQ.

	WFQ	CQ	PQ
Number of Queues	<ul style="list-style-type: none"> Configurable number of queues (256 user queues, by default) 	<ul style="list-style-type: none"> 16 user queues 	<ul style="list-style-type: none"> 4 queues
Kind of Service	<ul style="list-style-type: none"> Ensures fairness among all traffic flows based on weights 	<ul style="list-style-type: none"> Round-robin service Proportional allocation of bandwidth for different classes of service 	<ul style="list-style-type: none"> High priority queues serviced first Absolute prioritization; ensures critical traffic of highest priority
Configuration	<ul style="list-style-type: none"> No configuration required 	<ul style="list-style-type: none"> Requires configuration 	<ul style="list-style-type: none"> Requires configuration

First-In, First-Out Queueing

In its simplest form, FIFO queueing—also known as first-come, first-served (FCFS) queueing—involves storing packets when the network is congested and forwarding them in order of arrival when the network is no longer congested.

FIFO embodies no concept of priority or classes of traffic and consequently makes no decision about packet priority. There is only one queue, and all packets are treated equally. Packets are sent out an interface in the order in which they arrive. Higher priority packets are not transmitted faster than lower priority packets.

When FIFO is used, ill-behaved sources can consume all the bandwidth, bursty sources can cause delays in time-sensitive or important traffic, and important traffic can be dropped because less important traffic fills the queue.

When no other queueing strategies are configured, all interfaces excepting serial interfaces at E1 (2.048 Mbps) and below use FIFO by default. (Serial interfaces at E1—2.048 Mbps—and below use WFQ by default.)

FIFO, which is the fastest method of queueing, is effective for large links that have little delay and minimal congestion. If your link has very little congestion, FIFO queueing may be the only queueing you need to use.

Weighted Fair Queueing

This section discusses these two types of WFQ:

- Weighted Fair Queueing
- VIP-Distributed Weighted Fair Queueing

Table 6 summarizes the differences between WFQ and VIP-Distributed WFQ (DWFQ).

Table 6 **WFQ and DWFQ Comparison**

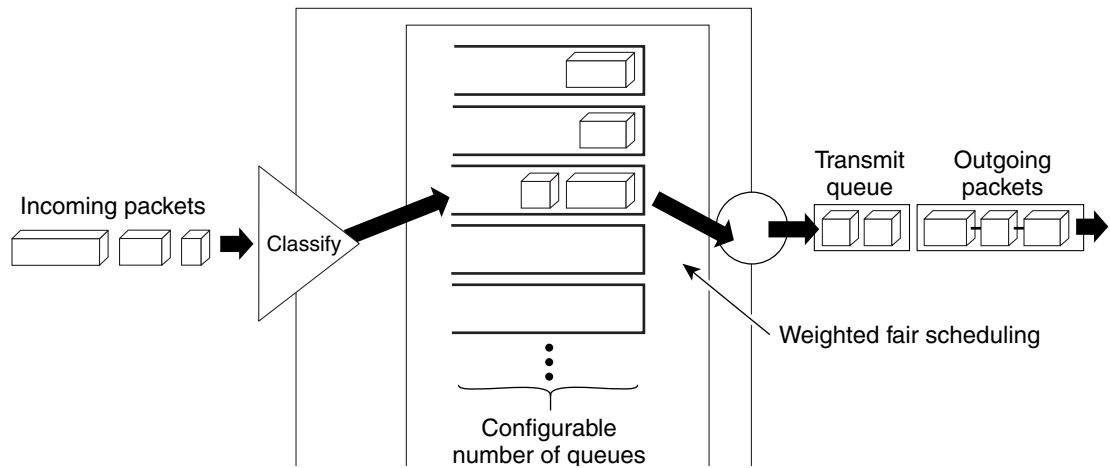
WFQ	VIP-Distributed WFQ
Flow-based WFQ	Flow-based WFQ
<ul style="list-style-type: none"> • Weighted, when packets are classified • FQ, when packets are not classified 	<ul style="list-style-type: none"> • FQ, not weighted
Runs on all standard IOS platforms	Runs on Versatile Interface Processor (faster performance)

All queueing is transacted by the Versatile Interface Processor (VIP). On the VIP, all packets are transmitted directly out the interface. An RSP resides on the same platform as the VIP. The RSP handles all tasks associated with system maintenance and routing. The VIP and the RSP each handle some scheduling. The dual-processor support accounts for the faster speed of VIP-Distributed WFQ over WFQ running on standard IOS platforms.

Weighted Fair Queueing

WFQ is an automated scheduling method that provides fair bandwidth allocation to all network traffic. WFQ applies priority, or weights, to identified traffic to classify traffic into conversations and determine how much bandwidth each conversation is allowed relative to other conversations. WFQ is a flow-based algorithm that simultaneously schedules interactive traffic to the front of a queue to reduce response time and fairly shares the remaining bandwidth among high-bandwidth flows. In other words, WFQ allows you to give low-volume traffic, such as Telnet sessions, priority over high-volume traffic, such as FTP sessions. WFQ gives concurrent file transfers balanced use of link capacity; that is, when multiple file transfers occur, the transfers are given comparable bandwidth. Figure 5 shows how WFQ works.

Figure 5 Weighted Fair Queueing



WFQ overcomes a serious limitation of FIFO queueing. When FIFO is in effect, traffic is transmitted in the order received without regard for bandwidth consumption or the associated delays. As a result, file transfers and other high-volume network applications often generate series of packets of associated data. These related packets are known as packet trains. Packet trains are groups of packets that tend to move together through the network. These packet trains can consume all available bandwidth, depriving other traffic of it.

WFQ provides traffic priority management that dynamically sorts traffic into messages that make up a conversation. WFQ breaks up the train of packets within a conversation to ensure that bandwidth is shared fairly between individual conversations and that low-volume traffic is transferred in a timely fashion.

WFQ classifies traffic into different flows based on packet header addressing, including such characteristics as source and destination network or MAC address, protocol, source and destination port and socket numbers of the session, Frame Relay data-link connection identifier (DLCI) value, and type of service (ToS) value. There are two categories of flows: high-bandwidth sessions and low-bandwidth sessions. Low-bandwidth traffic has effective priority over high-bandwidth traffic, and high-bandwidth traffic shares the transmission service proportionally according to assigned weights. Low-bandwidth traffic streams, which comprise the majority of traffic, receive preferential service, transmitting their entire offered loads in a timely fashion. High-volume traffic streams share the remaining capacity proportionally among themselves.

WFQ places packets of the various conversations in the fair queues before transmission. The order of removal from the fair queues is determined by the virtual time of the delivery of the last bit of each arriving packet.

New messages for high-bandwidth flows are discarded after the congestive-messages threshold has been met. However, low-bandwidth flows, which include control-message conversations, continue to enqueue data. As a result, the fair queue may occasionally contain more messages than are specified by the threshold number.

WFQ can manage duplex data streams, such as those between pairs of applications, and simplex data streams such as voice or video.

The WFQ algorithm also addresses the problem of round-trip delay variability. If multiple high-volume conversations are active, their transfer rates and interarrival periods are made much more predictable. WFQ greatly enhances algorithms such as SNA Logical Link Control (LLC) and Transmission Control Protocol (TCP) congestion control and slow start features.

WFQ is used as the default queuing mode on most serial interfaces configured to run at or below E1 speeds (2.048 Mbps).

WFQ provides the solution for situations in which it is desirable to provide consistent response time to heavy and light network users alike without adding excessive bandwidth. WFQ automatically adapts to changing network traffic conditions.

Restrictions

WFQ is not supported presently with tunneling and encryption because these features modify the packet content information required by WFQ for classification.

WFQ and IP Precedence

WFQ is IP Precedence-aware. That is, it is able to detect higher priority packets marked with precedence by the IP Forwarder and can schedule them faster, providing superior response time for this traffic. Thus, as the precedence increases, WFQ allocates more bandwidth to the conversation during periods of congestion.

WFQ assigns a weight to each flow, which determines the transmit order for queued packets. In this scheme, lower weights are served first. For standard IOS WFQ, the IP Precedence serves as a divisor to this weighting factor.

Like CQ, WFQ transmits a certain number of bytes from each queue. With WFQ, each queue corresponds to a different flow. For each cycle through all the flows, WFQ effectively transmits a number of bytes equal to the precedence of the flow plus one.

Actually, this number is only used as a ratio to determine how many bytes/packets to transmit. However, for the purposes of understanding WFQ, using this number as the byte count is sufficient. For instance, traffic with an IP Precedence field value of 7 gets a lower weight than traffic with an IP Precedence field value of 3, thus, the priority in transmit order.

To determine the bandwidth allocation for each queue, divide the byte count for the flow by the total byte count for all flows.

For example, if you have one flow at each precedence level, each flow will get precedence+1 parts of the link:

$$1+2+3+4+5+6+7+8 = 36$$

Thus, precedence 0 traffic will get 1/36 of the bandwidth, precedence 1 traffic will get 2/36, and precedence 7 traffic will get 8/36.

However, if you have 18 precedence 1 flows and one of each of the rest, the total is now:

$$1+2(18)+3+4+5+6+7+8=70$$

Precedence 0 traffic will get 1/70, each of the precedence 1 flows will get 2/70, and so on.

As flows are added or ended, the actual allocated bandwidth will continuously change.

WFQ and Resource Reservation Protocol (RSVP)

RSVP uses WFQ to allocate buffer space and schedule packets, and guarantee bandwidth for reserved flows. WFQ works with RSVP to help provide differentiated and guaranteed QoS services.

RSVP is the IETF Internet Standard (RFC 2205) protocol for allowing an application to dynamically reserve network bandwidth. RSVP enables applications to request a specific QoS for a data flow. Cisco's implementation allows RSVP to be initiated within the network using configured proxy RSVP.

RSVP is the only standard signaling protocol designed to guarantee network bandwidth from end to end for IP networks. Hosts and routers use RSVP to deliver QoS requests to the routers along the paths of the data stream and to maintain router and host state to provide the requested service, usually bandwidth and latency. RSVP uses a mean data rate, the largest amount of data the router will keep in queue, and minimum QoS to determine bandwidth reservation.

WFQ or WRED acts as the preparator for RSVP, setting up the packet classification and scheduling required for the reserved flows. Using WFQ, RSVP can deliver an Integrated Services Guaranteed Service.

WFQ and Frame Relay

WFQ weights are affected by Frame Relay discard eligible (DE), forward explicit congestion notification (FECN), and backward explicit congestion notification (BECN) bits when traffic is switched by the Frame Relay switching module. Once congestion is flagged, the weights used by the algorithm are altered so that the conversation encountering the congestion transmits less frequently.

Considerations

Although WFQ automatically adapts to changing network traffic conditions, it does not offer the degree of precision control over bandwidth allocation that CQ does.

VIP-Distributed Weighted Fair Queueing

VIP-Distributed WFQ is a special high-speed version of WFQ that runs on the VIP. It is supported on the following routers with a VIP2-40 or greater interface processor:

- Cisco 7000 series with RSP7000
- Cisco 7500 series

A VIP2-50 interface processor is recommended when the aggregate line rate of the port adapters on the VIP is greater than DS3. A VIP2-50 card is required for OC-3 rates.

To use VIP-Distributed WFQ, Distributed Cisco Express Forwarding (DCEF) switching must be enabled on the interface. For more information on CEF, refer to the *Cisco IOS Switching Services Configuration Guide* and the *Cisco IOS Switching Services Command Reference*.

This section describes flow-based VIP-Distributed WFQ. This section also includes a discussion that describes the drop policy used by both types of VIP-Distributed WFQ.

Note The VIP-Distributed WFQ implementation differs from WFQ that runs on all other platforms.

Flow-Based VIP-Distributed Weighted Fair Queueing

With flow-based VIP-Distributed WFQ, packets are classified by flow. Packets with the same source IP address, destination IP address, source TCP or UDP port, destination TCP or UDP port, protocol, and ToS field belong to the same flow. (All non-IP packets are treated as flow 0.)

Each flow corresponds to a separate output queue. When a packet is assigned to a flow, it is placed in the queue for that flow. During periods of congestion, VIP-Distributed WFQ allocates an equal share of the bandwidth to each active queue.

Flow-based VIP-Distributed WFQ is also called fair queueing because all flows are equally weighted and allocated equal bandwidth. In the current implementation of VIP-Distributed WFQ, weights are not assigned to flows. With VIP-Distributed WFQ, well-behaved hosts are protected from badly behaved hosts.

Restrictions

Use VIP-Distributed WFQ with IP traffic. All non-IP traffic is treated as a single flow and, therefore, placed in the same queue.

VIP-Distributed WFQ has the following restrictions:

- Can be configured on interfaces, but not subinterfaces.
- Is not supported on Fast EtherChannel, tunnel interfaces, or other logical (virtual) interfaces such as MLPPP.
- Cannot be configured on the same interface as RSP-based PQ, CQ, or WFQ.

Drop Policy

VIP-Distributed WFQ keeps track of the number of packets in each queue and the total number of packets in all queues.

When the total number of packets is below the aggregate limit, queues can buffer more packets than the individual queue limit.

When the total number of packets reaches the aggregate limit, the interface starts enforcing the individual queue limits. Any new packets that arrive for a queue that has exceeded its individual queue limit are dropped. Packets that are already in the queue will not be dropped, even if the queue is over the individual limit.

In some cases, the total number of packets in all queues put together may exceed the aggregate limit.

Custom Queueing

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

How It Works

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

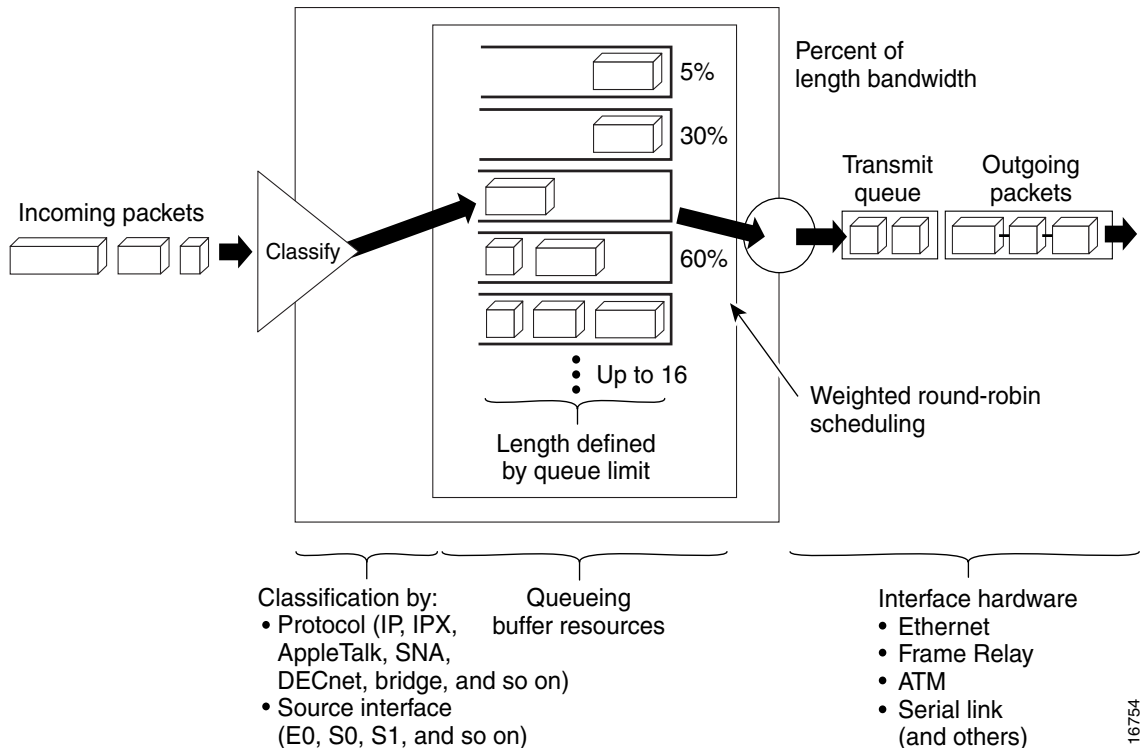
When CQ is enabled on an interface, the system maintains 17 output queues for that interface. You can specify queues 1 through 16. Associated with each output queue is a configurable byte count, which specifies how many bytes of data the system should deliver from the current queue before it moves on to the next queue.

Queue number 0 is a system queue; it is emptied before any of the queues numbered 1 through 16 are processed. The system queues high priority packets, such as keepalive packets and signaling packets, to this queue. Other traffic cannot be configured to use this queue.

For queue numbers 1 through 16, the system cycles through the queues sequentially (in a round-robin fashion), dequeuing the configured byte count from each queue in each cycle, delivering packets in the current queue before moving on to the next one. When a particular queue is being processed, packets are sent until the number of bytes sent exceeds the queue byte count or the queue is empty. Bandwidth used by a particular queue can only be indirectly specified in terms of byte count and queue length.

Figure 6 shows how CQ behaves.

Figure 6 Custom Queueing



16754

CQ ensures that no application or specified group of applications achieves more than a predetermined proportion of overall capacity when the line is under stress. Like PQ, CQ is statically configured and does not automatically adapt to changing network conditions.

On most platforms, all protocols are classified in the fast switching path.

Determining Byte Count Values for Queues

In order to allocate bandwidth to different queues, you must specify the byte count for each queue.

How the Byte Count Is Used

The router sends packets from a particular queue until the byte count is exceeded. Once the byte count value is exceeded, the packet that is currently being transmitted will be completely sent. Therefore, if you set the byte count to 100 bytes and the packet size of your protocol is 1024 bytes, then every time this queue is serviced, 1024 bytes will be sent, not 100 bytes.

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

Thus, setting the byte count too low can result in an unintended bandwidth allocation.

However, very large byte counts will produce a “jerky” distribution. That is, if you assign 10 KB, 10 KB, and 10 KB to three queues in the example given, each protocol is serviced promptly when its queue is the one being serviced, but it may be a long time before the queue is serviced again. A better solution is to specify 500-byte, 600-byte, and 500-byte counts for the queue. This configuration results in a ratio of 31/38/31, which may be acceptable.

In order to service queues in a timely manner and ensure that the configured bandwidth allocation is as close as possible to the required bandwidth allocation, you must determine the byte count based on each protocol’s packet size, otherwise your percentages may not match what you configure.

Note Some protocols, such as IPX, will negotiate the frame size at session startup time.

Determining the Byte Count

To determine the correct byte counts, perform the following tasks:

Step 1 For each queue, divide the percentage of bandwidth you want to allocate to the queue by the packet size, in bytes. For example, assume the packet size for protocol A is 1086 bytes, protocol B is 291 bytes, and protocol C is 831 bytes. We want to allocate 20 percent for A, 60 percent for B, and 20 percent for C. The ratios would be:

$20/1086, 60/291, 20/831$ or

$0.01842, 0.20619, 0.02407$

Step 2 Normalize the numbers by dividing by the lowest number:

$1, 11.2, 1.3$

The result is the ratio of the number of packets that must be sent so that the percentage of bandwidth that each protocol uses is approximately 20, 60, and 20 percent.

Step 3 A fraction in any of the ratio values means that an additional packet will be sent. Round up the numbers to the next whole number to obtain the actual packet count.

In this example, the actual ratio will be 1 packet, 12 packets, and 2 packets.

- Step 4** Convert the packet number ratio into byte counts by multiplying each packet count by the corresponding packet size.

In this example, the number of packets sent is one 1086-byte packet, twelve 291-byte packets, and two 831-byte packets or 1086, 3492, and 1662 bytes, respectively, from each queue. These are the byte counts you would specify in your custom queueing configuration.

- Step 5** To determine the bandwidth distribution this ratio represents, first determine the total number of bytes sent after all three queues are serviced:

$$(1 \times 1086) + (12 \times 291) + (2 \times 831) = 1086 + 3492 + 1662 = 6240$$

- Step 6** Then determine the percentage of the total number of bytes sent from each queue:

$$1086/6240, 3492/6240, 1662/6240 = 17.4, 56, \text{ and } 26.6 \text{ percent}$$

As you can see, this is close to the desired ratio of 20/60/20.

- Step 7** If the actual bandwidth is not close enough to the desired bandwidth, multiply the original ratio of 1:11.2:1.3 by the best value, trying to get as close to three integer values as possible. Note that the multiplier you use need not be an integer. For example, if we multiply the ratio by two, we get 2:22.4:2.6. We would now send two 1086-byte packets, twenty-three 291-byte packets, and three 831-byte packets, or 2172/6693/2493, for a total of 11,358 bytes. The resulting ration is 19/59/22 percent, which is much closer to the desired ratio that we achieved.

Window Size

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

Therefore, with a window size of one, only one frame will be sent each time. If your frame count is set to 2 KB, and your frame size is 256 bytes, then only 256 bytes will be sent each time this queue is serviced.

Why Use Custom Queueing?

You can use the Cisco IOS QoS CQ feature to provide specific traffic guaranteed bandwidth at a potential congestion point, assuring the traffic a fixed portion of available bandwidth and leaving the remaining bandwidth to other traffic. For example, you could reserve half of the bandwidth for SNA data, allowing the remaining half to be used by other protocols.

If a particular type of traffic is not using the bandwidth reserved for it, then unused bandwidth can be dynamically allocated to other traffic types.

Considerations

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

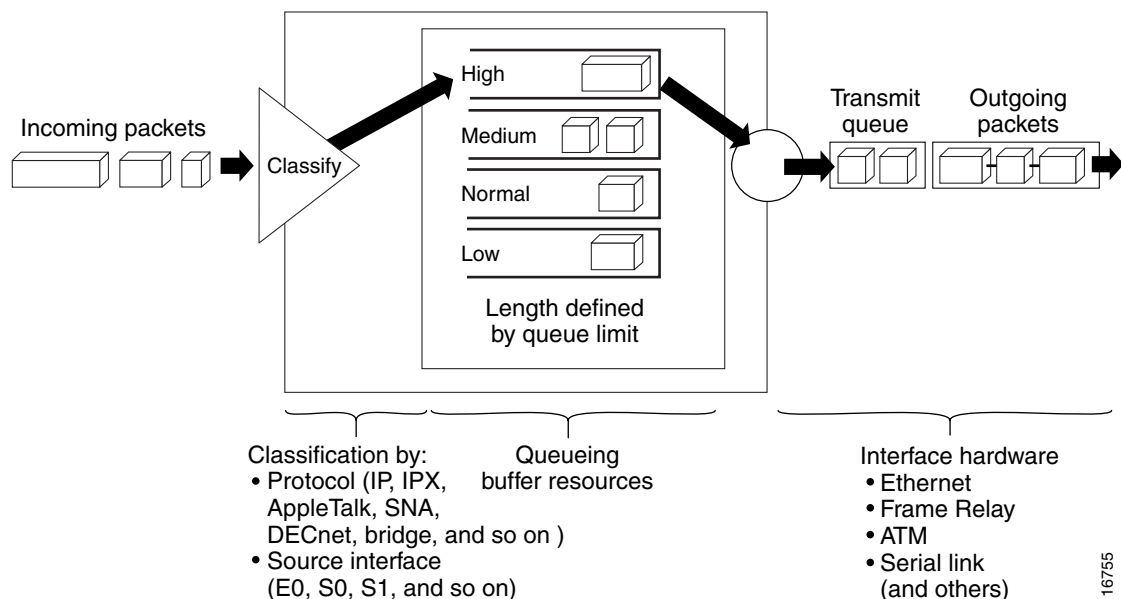
Priority Queueing

PQ allows you to define how traffic is prioritized in the network. You configure four traffic priorities. You can define a series of filters based on packet characteristics to cause the router to place traffic into these four queues; the queue with the highest priority is serviced first until it is empty, then the lower queues are serviced in sequence.

How It Works

During transmission, PQ gives priority queues absolute preferential treatment over low priority queues; important traffic, given the highest priority, always takes precedence over less important traffic. Packets are classified based on user-specified criteria and placed into one of the four output queues—high, medium, normal, and low—based on the assigned priority. Packets that are not classified by priority fall into the normal queue. Figure 7 illustrates this process.

Figure 7 Priority Queueing



When a packet is to be sent out an interface, the priority queues on that interface are scanned for packets in descending order of priority. The high priority queue is scanned first, then the medium priority queue, and so on. The packet at the head of the highest queue is chosen for transmission. This procedure is repeated every time a packet is to be transmitted.

The maximum length of a queue is defined by the length limit. When a queue is longer than the queue limit, all additional packets are dropped.

Note The priority output queueing mechanism can be used to manage traffic from all networking protocols. Additional fine-tuning is available for IP and for setting boundaries on the packet size.

How Packets Are Classified for Priority Queueing

A priority list is a set of rules that describe how packets should be assigned to priority queues. A priority list might also describe a default priority or the queue size limits of the various priority queues.

Packets can be classified by the following:

- Protocol or subprotocol type
- Incoming interface
- Packet size
- Fragments
- Access list

Keepalives sourced by the network server are always assigned to the high priority queue; all other management traffic (such as IGRP updates) must be configured. Packets that are not classified by the priority list mechanism are assigned to the normal queue.

Why Use Priority Queueing?

PQ provides absolute preferential treatment to high priority traffic, ensuring that mission-critical traffic traversing various WAN links gets priority treatment. In addition, PQ provides a faster response time than do other methods of queueing.

Although you can enable priority output queueing for any interface, it is best used for low-bandwidth, congested serial interfaces.

Considerations

When choosing to use PQ, consider that because lower priority traffic is often denied bandwidth in favor of higher priority traffic, use of PQ could, in the worst case, result in lower priority traffic never being transmitted. To avoid inflicting these conditions on lower priority traffic, you can use traffic shaping or CAR to rate-limit the higher priority traffic.

PQ introduces extra overhead that is acceptable for slow interfaces, but may not be acceptable for higher speed interfaces such as Ethernet. With PQ enabled, the system takes longer to switch packets because the packets are classified by the processor card.

PQ uses a static configuration and does not adapt to changing network conditions.

Restrictions

PQ is not supported on any tunnels.