

# Low Latency Queueing

---

This feature module describes the Low Latency Queueing feature. It includes information on the benefits of the new feature, supported platforms, related documents, and so forth.

This document includes the following sections:

- Feature Overview, page 1
- Supported Platforms, page 4
- Supported Standards, MIBs, and RFCs, page 5
- Prerequisites, page 5
- Configuration Tasks, page 5
- Monitoring and Maintaining Low Latency Queueing, page 6
- Configuration Examples, page 6
- Command Reference, page 9
- Glossary, page 12

## Feature Overview

The Low Latency Queueing feature brings strict priority queueing to Class-Based Weighted Fair Queueing (CBWFQ). Strict priority queueing allows delay-sensitive data such as voice to be dequeued and sent first (before packets in other queues are dequeued), giving delay-sensitive data preferential treatment over other traffic.

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

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

The Low Latency Queueing feature provides strict priority queueing for CBWFQ, reducing jitter in voice conversations. Configured by the **priority** command, Low Latency Queueing enables use of a single, strict priority queue within CBWFQ at the class level, allowing you to direct traffic belonging

to a class to the CBWFQ strict priority queue. To enqueue class traffic to the strict priority queue, you configure the **priority** command for the class after you specify the named class within a policy map. (Classes to which the **priority** command is applied are considered priority classes.) Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

One of the ways in which the strict priority queuing used within CBWFQ differs from its use outside CBWFQ is in the parameters it takes. Outside CBWFQ, by using the **ip rtp priority** command, you specify the range of UDP ports whose voice traffic flows are to be given priority service. Using the **priority** command, because you can configure the priority status for a class within CBWFQ, you are no longer limited to a UDP port number to stipulate priority flows. Instead, all of the valid match criteria used to specify traffic for a class now applies to priority traffic. These methods of specifying traffic for a class include matching on access lists, protocols, and input interfaces. Moreover, within an access list you can specify that traffic matches are allowed based on the IP Differentiated Services Code Point (DSCP) value that is set using the first six bits of the Type of Service (ToS) byte in the IP header.

Although it is possible to enqueue various types of real-time traffic to the strict priority queue, we strongly recommend that you direct only voice traffic to it. This recommendation is made because voice traffic is well-behaved, whereas other types of real-time traffic are not. Moreover, voice traffic requires that delay be nonvariable in order to avoid jitter. Real-time traffic such as video could introduce variation in delay, thereby thwarting the steadiness of delay required for successful voice traffic transmission.

## Guaranteed Bandwidth

When you specify the **priority** command for a class, it takes a bandwidth argument that gives maximum bandwidth in kilobits per second (kbps). You use this parameter to specify the maximum amount of bandwidth allocated for packets belonging to the class configured with the **priority** command. The bandwidth parameter both guarantees bandwidth to the priority class and restrains the flow of packets from the priority class.

In the event of congestion, when the bandwidth is exceeded policing is used to drop packets. Voice traffic enqueued to the priority queue is UDP-based and therefore not adaptive to the early packet drop characteristic of Weighted Random Early Detection (WRED). Because WRED is ineffective, you cannot use the WRED **random-detect** command with the **priority** command. In addition, because policing is used to drop packets and queue limit is not imposed, the **queue-limit** command cannot be used with the **priority** command.

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

Priority traffic metering has the following qualities:

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

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

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

The bandwidth allocated for a priority queue always includes the Layer 2 encapsulation header. However, it does not include other headers, such as ATM cell tax overheads. When you calculate the amount of bandwidth to allocate for a given priority class, you must account for the fact the Layer 2 headers are included. When ATM is used, you must account for the fact that ATM cell tax overhead is not included. You must also allow bandwidth for the possibility of jitter introduced by routers in the voice path.

Consider this case that uses ATM. Suppose a voice stream of 60 bytes emitting 50 packets per second is encoded using G.729. Prior to converting the voice stream to cells, the meter for the priority queue used for the voice stream assesses the length of the packet after the Layer 2 LLC headers have been added.

Given the 8-byte Layer 2 LLC header, the meter will take into account a 68-byte packet. Because ATM cells are a standard 53 bytes long, before the 68-byte packet is emitted on the line, it is divided into two 53-byte ATM cells. Thus, the bandwidth consumed by this flow is 106 bytes per packet.

For this case, then, you must configure the bandwidth to be at least 27.2 kbps ( $68 * 50 * 8 = 27.2$  kbps). However, recall that you must also allow for the cell tax overhead, which is not accounted for by the configured bandwidth. In other words, the sum of the bandwidths for all classes must be less than the interface bandwidth by at least  $(106 - 68) * 50 * 8 = 15.2$  kbps. You should also remember to allow bandwidth for router-introduced jitter.

## Benefits

### Provides Strict Priority Service on ATM VCs and Serial Interfaces

The strict priority queueing scheme allows delay-sensitive data such as voice to be dequeued and sent first—that is, before packets in other queues are dequeued. Delay-sensitive data is given preferential treatment over other traffic. This feature provides strict priority queueing on ATM virtual circuits (VCs); the IP RTP Priority feature only allows priority queueing on interfaces.

### Not Limited to UDP Port Numbers

Because you can configure the priority status for a class within CBWFQ, you are no longer limited to UDP port numbers to stipulate priority flows. Instead, all of the valid match criteria used to specify traffic for a class now applies to priority traffic.

### Admission Control

By configuring the maximum amount of bandwidth allocated for packets belonging to a class, you can avoid starving non-priority traffic.

## Restrictions

- The Low Latency Queueing feature supports Voice over IP (VoIP) on serial links and ATM PVCs. It does not support VoIP over Frame Relay links.
- If you use access lists to configure matching port numbers, this feature provides priority matching for all port numbers (as compared to the IP RTP Priority feature, which only provides priority for even port numbers). Because voice typically exists on even port numbers, and control packets are generated on odd port numbers, control packets are also given priority when using this feature. On very slow links, giving priority to both voice and control packets may produce degraded voice quality. Therefore, if you are only assigning priority based on port numbers, you should use the **ip rtp priority** command instead of the **priority** command.
- The **random-detect**, **queue-limit**, and **bandwidth** commands cannot be used while the **priority** command is configured.
- The **priority** command can be configured in multiple classes, but it should only be used for voice-like, constant bit rate (CBR) traffic. If the traffic is not CBR, you must configure a large enough *bandwidth* parameter to absorb the data bursts.

## Related Features and Technologies

The Low Latency Queueing feature is related to the following features:

- CBWFQ
- IP RTP Priority
- Link Fragmentation and Interleaving (LFI)
- Priority Queueing (PQ)
- Weighted Fair Queueing (WFQ)

## Related Documents

- *Quality of Service Solutions Configuration Guide*, Cisco IOS Release 12.0
- *Quality of Service Solutions Command Reference*, Cisco IOS Release 12.0
- *Class-Based Weighted Fair Queueing* feature module
- *IP RTP Priority* feature module

## Supported Platforms

- Cisco 1003
- Cisco 1004
- Cisco 1005
- Cisco 1600 series
- Cisco 2500 series
- Cisco 2600 series
- Cisco 3600 series

- Cisco 3800 series
- Cisco 4000 series (Cisco 4000, 4000-M)
- Cisco 5200 series
- Cisco 7000 series
- Cisco 7200 series
- Cisco 7500/RSP series
- Cisco AS5300

## Supported Standards, MIBs, and RFCs

### Standards

None

### MIBs

This feature supports no new MIBs.

For descriptions of supported MIBs and how to use MIBs, see the Cisco MIB web site on CCO at <http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml>.

### RFCs

None

## Prerequisites

To use this feature, you should be familiar with the following:

- Access control lists
- ATM permanent virtual circuits (PVCs)
- Bandwidth management
- CBWFQ
- LFI
- Virtual templates and virtual access interfaces

## Configuration Tasks

See the following sections for configuration tasks for the Low Latency Queueing feature. Each task in the list indicates if the task is optional or required.

- Configuring Low Latency Queueing (Required)
- Verifying Low Latency Queueing (Optional)

### Configuring Low Latency Queueing

To give priority to a class within a policy map, use the following command in policy-map class configuration mode:

Command	Purpose
Router (config-pmap-c) # <b>priority</b> <i>bandwidth</i>	Reserves a strict priority queue for CBWFQ traffic.

### Verifying Low Latency Queueing

To see the contents of the priority queue (such as queue depth and the first packet queued), use the following command in EXEC mode:

Command	Purpose
Router# <b>show queue</b> <i>interface-type interface-number</i>	Lists fair queueing configuration and statistics for a particular interface.

The priority queue is the queue whose conversation ID is equal to the number of dynamic queues plus 8. The packets in the priority queue have a weight of 0.

## Monitoring and Maintaining Low Latency Queueing

To tune your RTP bandwidth or decrease RTP traffic if the priority queue is experiencing drops, use one or more of the following commands:

Command	Purpose
Router# <b>debug priority</b>	Displays priority queueing output if packets are dropped from the priority queue.
Router# <b>show queue</b> <i>interface-type interface-number</i>	Lists fair queueing configuration and statistics for a particular interface.
Router# <b>show policy interface</b> <i>interface-name</i>	Displays the configuration of all classes configured for all service policies on the specified interface. Shows if packets and bytes were discarded or dropped for the priority class in the service policy attached to the interface.

### Configuration Examples

This section provides the following configuration examples:

- ATM PVC Configuration
- Virtual Template Configuration
- Multilink Bundle Configuration

## ATM PVC Configuration

In the following example, a strict priority queue (with a guaranteed allowed bandwidth of 50 kbps) is reserved for traffic that is sent from the source address (10.10.10.10) to the destination address (10.10.10.20), in the range of ports 16384 through 20000 and 53000 through 56000.

First, the following commands configure access list 102 to match the desired voice traffic:

```
router(config)# access-list 102 permit udp host 10.10.10.10 host 10.10.10.20 range
16384 20000
router(config)# access-list 102 permit udp host 10.10.10.10 host 10.10.10.20 range
53000 56000
```

Next, the class map voice is defined, and the policy map policy1 is created; a strict priority queue for the class voice is reserved, a bandwidth of 20 kbps is configured for the class bar, and the default class is configured for WFQ. The **service-policy** command then attaches the policy map to PVC 0/102 on subinterface atm1/0:

```
router(config)# class-map voice
router(config-cmap)# match access-group 102

router(config)# policy-map policy1
router(config-pmap)# class voice
router(config-pmap-c)# priority 50
router(config-pmap)# class bar
router(config-pmap-c)# bandwidth 20
router(config-pmap)# class class-default
router(config-pmap-c)# fair-queue

router(config)# interface atm1/0
router(config-subif)# pvc 0/102
router(config-subif-vc)# service-policy output policy1
```

## Virtual Template Configuration

The following example configures a strict priority queue in a virtual template configuration with CBWFQ. Traffic on virtual template 1 that is matched by access list 102 will be directed to the strict priority queue.

First, the class map voice is defined, and the policy map policy1 is created. A strict priority queue (with a guaranteed allowed bandwidth of 50 kbps) is reserved for the class voice.

```
router(config)# class-map voice
router(config-cmap)# match access-group 102
router(config)# policy-map policy1
router(config-pmap)# class voice
router(config-pmap-c)# priority 50
```

Next, the **service-policy** command attaches the policy map policy1 to the virtual template 1:

```

router(config)# multilink virtual-template 1
router(config)# interface virtual-template 1
router(config-if)# ip address 172.16.1.1 255.255.255.0
router(config-if)# no ip directed-broadcast
router(config-if)# service-policy output policy1
router(config-if)# ppp multilink
router(config-if)# ppp multilink fragment-delay 20
router(config-if)# ppp multilink interleave
router(config-if)# end

router(config)# interface serial 2/0
router(config-if)# bandwidth 256
router(config-if)# no ip address
router(config-if)# no ip directed-broadcast
router(config-if)# encapsulation ppp
router(config-if)# no fair-queue
router(config-if)# clockrate 256000
router(config-if)# ppp multilink

```

## Multilink Bundle Configuration

The following example configures a strict priority queue in a multilink bundle configuration with CBWFQ. Traffic on serial interface 2/0 that is matched by access list 102 will be directed to the strict priority queue. The advantage to using multilink bundles is that you can specify different **priority** parameters on different interfaces. To specify different **priority** parameters, you would configure two multilink bundles with different parameters.

First, the class map voice is defined, and the policy map policy1 is created. A strict priority queue (with a guaranteed allowed bandwidth of 50 kbps) is reserved for the class voice.

```

router(config)# class-map voice
router(config-cmap)# match access-group 102
router(config)# policy-map policy1
router(config-pmap)# class voice
router(config-pmap-c)# priority 50

```

The following commands create multilink bundle 1. The policy1 policy map is attached to the bundle by the **service-policy** command.

```

router(config)# interface multilink 1
router(config-if)# ip address 172.17.254.161 255.255.255.248
router(config-if)# no ip directed-broadcast
router(config-if)# no ip mroute-cache
router(config-if)# service-policy output policy1
router(config-if)# ppp multilink
router(config-if)# ppp multilink fragment-delay 20
router(config-if)# ppp multilink interleave

```

In the next part of the example, the **multilink-group** command configures serial interface 2/0 to be part of multilink bundle 1, which effectively directs traffic on serial interface 2/0 that is matched by access list 102 to the strict priority queue.

```
router(config)# interface serial 2/0
router(config-if)# bandwidth 256
router(config-if)# no ip address
router(config-if)# no ip directed-broadcast
router(config-if)# encapsulation ppp
router(config-if)# no fair-queue
router(config-if)# clockrate 256000
router(config-if)# ppp multilink
router(config-if)# multilink-group 1
```

## Command Reference

This section documents the new **priority** command that configures the Low Latency Queueing feature. All other commands used with this feature are documented in the Cisco IOS Release 12.0 command reference publications or Cisco IOS Release 12.0 T feature module publications.

## priority

To give priority to a class within a policy map, use the **priority** policy-map class configuration command. To disable the strict priority queue, use the **no** form of this command.

```
priority bandwidth
no priority [bandwidth]
```

### Syntax Description

<i>bandwidth</i>	Guaranteed allowed bandwidth (in kbps) for the priority traffic. Beyond the guaranteed bandwidth, the priority traffic will be dropped in the event of congestion to ensure that the nonpriority traffic is not starved.
------------------	--

### Defaults

No default behavior or values.

### Command Modes

Policy-map class configuration

### Command History

Release	Modification
12.0(6)T	This command was introduced.

### Usage Guidelines

This command provides strict priority queueing for CBWFQ. Strict priority queueing allows delay-sensitive data such as voice to be dequeued and sent first (before packets in other queues are dequeued), giving delay-sensitive data preferential treatment over other traffic.

The **priority** command allows you to set up classes based on a variety of criteria (not just UDP ports) and assign priority to them, and is available for use on serial interfaces and ATM PVCs. A similar command, **ip rtp priority**, allows you to stipulate priority flows based only on UDP port numbers and is not available for ATM PVCs.

The *bandwidth* argument is used to specify the maximum amount of bandwidth allocated for packets belonging to a class configured with the **priority** command. The bandwidth parameter both guarantees bandwidth to the priority class and restrains the flow of packets from the priority class.

When the device is not congested, the priority class traffic is allowed to exceed its allocated bandwidth. When the device is congested, the priority class traffic above the allocated bandwidth is discarded.

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

- Layer 2 encapsulations are accounted for in the amount of bandwidth specified with the **priority** command. However, care must be taken to configure a bandwidth that has room for cell-tax overhead and possible jitter introduced by the routers in the voice path.
- The **priority** command can be used for voice over IP (VoIP) on serial links and ATM PVCs. It does not support VoIP over Frame Relay links.

- The **random-detect**, **queue-limit**, or **bandwidth** commands cannot be used while the **priority** command is configured.
- The **priority** command can be configured in multiple classes, but it should only be used for voice-like, constant bit rate (CBR) traffic. If the traffic is not CBR, you must configure a large enough *bandwidth* parameter to absorb the data bursts.

Configuring the **priority** command in multiple classes provides the ability to police the priority classes individually. For an example, refer to the following configuration:

```
policy-map policy1
  class voice1
    priority 24
  class voice2
    priority 48
  class data
    bandwidth 20
```

In this example, voice1 and voice2 classes of traffic go into the high priority queue and get strict priority queueing over data traffic. However, voice1 traffic will be rate-limited to 24 kbps and voice2 traffic will be rate-limited to 48 kbps. The classes will be individually rate-limited (and given first-in first-out [FIFO] treatment) even if they go into the same queue.

## Examples

The following example configures strict priority queueing with a guaranteed bandwidth of 50 kbps for the policy map “policy1”:

```
router(config)# policy-map policy1
router(config-pmap)# class voice
router(config-pmap-c)# priority 50
```

## Related Commands

Command	Description
<b>debug priority</b>	Displays priority queueing events.
<b>ip rtp priority</b>	Reserves a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports.
<b>ip rtp reserve</b>	Reserves a special queue for a set of RTP packet flows belonging to a range of UDP destination ports.
<b>max-reserved-bandwidth</b>	Changes the percent of interface bandwidth allocated for CBWFQ and IP RTP Priority.
<b>show policy interface</b>	Displays the configuration of all classes configured for all service policies on the specified interface.
<b>show queue</b>	Lists fair queueing configuration and statistics for a particular interface.

## Glossary

**CBWFQ**—Class-Based Weighted Fair Queuing. Extends the standard WFQ functionality to provide support for user-defined traffic classes.

**Class-Based Weighted Fair Queuing**—See CBWFQ.

**LLC**—logical link control. Higher of the two data link layer sublayers defined by the IEEE. The LLC sublayer handles error control, flow control, framing, and MAC-sublayer addressing. The most prevalent LLC protocol is IEEE 802.2, which includes both connectionless and connection-oriented variants.

**logical link control**—See LLC.

**PPP**—Point-to-Point Protocol. Successor to SLIP that provides router-to-router and host-to-network connections over synchronous and asynchronous circuits. Whereas SLIP was designed to work with IP, PPP was designed to work with several network layer protocols, such as IP, IPX, and ARA. PPP also has built-in security mechanisms, such as CHAP and PAP. PPP relies on two protocols: LCP and NCP.

**Random Early Detection**—See RED.

**RED**—Random Early Detection. A congestion avoidance mechanism that takes advantage of TCP's congestion control mechanism. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it will decrease its transmission rate until all the packets reach their destination, indicating that the congestion is cleared.

**RTP**—Real-Time Transport Protocol. One of the IPv6 protocols. RTP is designed to provide end-to-end network transport functions for applications sending real-time data such as audio, video, or simulation data over multicast or unicast network services. RTP provides services such as payload type identification, sequence numbering, time-stamping, and delivery monitoring to real-time applications.

**SNA**—Systems Network Architecture. Large, complex, feature-rich network architecture developed in the 1970s by IBM. Similar in some respects to the OSI reference model, but with a number of differences.

**UDP**—User Datagram Protocol. Connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC 768.

**weighted fair queuing**—See WFQ.

**WFQ**—weighted fair queuing. Congestion management algorithm that identifies conversations (in the form of traffic streams), separates packets that belong to each conversation, and ensures that capacity is shared fairly between these individual conversations. WFQ is an automatic way of stabilizing network behavior during congestion and results in increased performance and reduced retransmission.