



Configuring Weighted Fair Queueing

This chapter describes the tasks for configuring weighted fair queueing (WFQ), VIP-Distributed WFQ (DWFQ), and class-based WFQ (CBWFQ) and the following related features, which provide strict priority queueing within WFQ or CBWFQ:

- IP RTP Priority Queueing
- Frame Relay IP RTP Priority Queueing
- Low latency queueing (LLQ)

For complete conceptual information, see the section “Weighted Fair Queueing” in the chapter “Congestion Management Overview” in this book.

For a complete description of the QoS commands in this chapter, refer to the *Cisco IOS Quality of Service Solutions Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Flow-Based Weighted Fair Queueing Configuration Task List

WFQ provides traffic priority management that automatically sorts among individual traffic streams without requiring that you first define access lists (ACLs). WFQ can also manage duplex data streams such as those between pairs of applications, and simplex data streams such as voice or video. There are two categories of WFQ sessions: high bandwidth and low bandwidth. Low-bandwidth traffic has effective priority over high-bandwidth traffic, and high-bandwidth traffic shares the transmission service proportionally according to assigned weights.

When WFQ is enabled for an interface, new messages for high-bandwidth traffic streams are discarded after the configured or default congestive messages threshold has been met. However, low-bandwidth conversations, which include control message conversations, continue to enqueue data. As a result, the fair queue may occasionally contain more messages than its configured threshold number specifies.

With standard WFQ, packets are classified by flow. Packets with the same source IP address, destination IP address, source TCP or User Datagram Protocol (UDP) port, or destination TCP or UDP port belong to the same flow. WFQ allocates an equal share of the bandwidth to each flow. Flow-based WFQ is also called fair queueing because all flows are equally weighted.

The Cisco IOS software provides two forms of flow-based WFQ:

- Standard WFQ, which is enabled by default on all serial interfaces that run at or below 2 Mbps, and can run on all Cisco serial interfaces.

- VIP-Distributed WFQ, which runs only on Cisco 7000 series routers with a Route Switch Processor (RSP)-based RSP7000 interface processor or Cisco 7500 series routers with a Versatile Interface Processor (VIP)-based VIP2-40 or greater interface processor. (A VIP2-50 interface processor is strongly recommended when the aggregate line rate of the port adapters on the VIP is greater than DS3. A VIP2-50 interface processor is required for OC-3 rates.) For configuration information on DWFQ, see the section “VIP-Distributed Weighted Fair Queueing Configuration Task List” later in this chapter.

To configure flow-based WFQ, perform the tasks in the following sections. The first section is required; the other section is optional.

- Configuring WFQ (Required)
- Monitoring Fair Queueing (Optional)

Flow-based WFQ is supported on unavailable bit rate (UBR), variable bit rate (VBR), and available bit rate (ABR) ATM connections.

See the end of this chapter for the section “Flow-Based WFQ Configuration Examples.”

Configuring WFQ

To configure flow-based fair queueing on an interface, use the following command in interface configuration mode:

Command	Purpose
Router(config-if)# fair-queue [<i>congestive-discard-threshold</i> [<i>dynamic-queues</i> [<i>reservable-queues</i>]]]	Configures an interface to use fair queueing.

Flow-based WFQ uses a traffic data stream discrimination registry service to determine to which traffic stream a message belongs. See the table accompanying the description of the **fair-queue** (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the attributes of a message that are used to classify traffic into data streams.

Defaults are provided for the congestion threshold after which messages for high-bandwidth conversations are dropped, and for the number of dynamic and reservable queues; however, you can fine-tune your network operation by changing these defaults. See the tables accompanying the description of the **fair-queue** (WFQ) command in the *Cisco IOS Quality of Service Solutions Command Reference* for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface or ATM VC. These values do not apply for DWFQ.



Note

WFQ is the default queueing mode on interfaces that run at E1 speeds (2.048 Mbps) or below. It is enabled by default for physical interfaces that do not use Link Access Procedure, Balanced (LAPB), X.25, or Synchronous Data Link Control (SDLC) encapsulations. WFQ is not an option for these protocols. WFQ is also enabled by default on interfaces configured for Multilink PPP (MLP). However, if custom queueing or priority queueing is enabled for a qualifying link, it overrides fair queueing, effectively disabling it. Additionally, WFQ is automatically disabled if you enable autonomous or silicon switching.

Monitoring Fair Queueing

To monitor flow-based fair queueing services in your network, use one or more of the following commands in EXEC mode:

Command	Purpose
Router# show interfaces [<i>interface</i>]	Displays statistical information specific to an interface.
Router# show queue <i>interface-type interface-number</i>	Displays the contents of packets inside a queue for a particular interface or VC.
Router# show queueing fair	Displays status of the fair queueing configuration.

VIP-Distributed Weighted Fair Queueing Configuration Task List

To configure DWFQ, perform one of the following mutually-exclusive tasks:

- Configuring Flow-Based DWFQ
- Configuring QoS-Group-Based DWFQ
- Configuring ToS-Based DWFQ
- Monitoring DWFQ (Optional)

If you enable flow-based DWFQ and then enable class-based DWFQ (either QoS-group based or ToS-based), class-based DWFQ will replace flow-based DWFQ.

If you enable class-based DWFQ and then want to switch to flow-based DWFQ, you must disable class-based DWFQ using the **no fair-queue class-based** command before enabling flow-based DWFQ.

If you enable one type of class-based DWFQ and then enable the other type, the second type will replace the first.

DWFQ runs only on Cisco 7000 series routers with a Route Switch Processor-based RSP7000 interface processor or Cisco 7500 series routers with a VIP-based VIP2-40 or greater interface processor. (A VIP2-50 interface processor is strongly recommended when the aggregate line rate of the port adapters on the VIP is greater than DS3. A VIP2-50 interface processor is required for OC-3 rates.)

DWFQ can be configured on interfaces but not subinterfaces. It is not supported on Fast EtherChannel, tunnel, or other logical or virtual interfaces such as MLP.

Configuring Flow-Based DWFQ

To configure flow-based DWFQ, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	Router(config-if)# fair-queue	Enables flow-based DWFQ.
Step 2	Router(config-if)# fair-queue aggregate-limit <i>aggregate-packet</i>	(Optional) Sets the total number of buffered packets before some packets may be dropped. Below this limit, packets will not be dropped.
Step 3	Router(config-if)# fair-queue individual-limit <i>individual-packet</i>	(Optional) Sets the maximum queue size for individual per-flow queues during periods of congestion.

For flow-based DWFQ, 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, and protocol belong to the same flow.

In general, you should not change the aggregate or individual limit value from the default. Use the **fair-queue aggregate-limit** and **fair-queue individual-limit** commands only if you have determined that you would benefit from using different values, based on your particular situation.

Configuring QoS-Group-Based DWFQ

To configure QoS-group-based DWFQ, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	Router(config-if)# fair-queue qos-group	Enables QoS-group-based DWFQ.
Step 2	Router(config-if)# fair-queue qos-group <i>number</i> weight <i>weight</i>	For each QoS group, specifies the percentage of the bandwidth to be allocated to each class.
Step 3	Router(config-if)# fair-queue aggregate-limit <i>aggregate-packet</i>	(Optional) Sets the total number of buffered packets before some packets may be dropped. Below this limit, packets will not be dropped.
Step 4	Router(config-if)# fair-queue individual-limit <i>individual-packet</i>	(Optional) Sets the maximum queue size for every per-flow queue during periods of congestion.
Step 5	Router(config-if)# fair-queue qos-group <i>number</i> limit <i>class-packet</i>	(Optional) Sets the maximum queue size for a specific QoS group queue during periods of congestion.

In general, you should not change the aggregate, individual, or class limit value from the default. Use the **fair-queue aggregate-limit**, **fair-queue individual-limit**, and **fair-queue limit** commands only if you have determined that you would benefit from using different values, based on your particular situation.

Configuring ToS-Based DWFQ

To configure ToS-based DWFQ, use the following commands in interface configuration mode:

	Command	Purpose
Step 1	Router(config-if)# fair-queue tos	Enables ToS-based DWFQ
Step 2	Router(config-if)# fair-queue tos number weight <i>weight</i>	(Optional) For each ToS class, specifies the percentage of the bandwidth to be allocated to each class.
Step 3	Router(config-if)# fair-queue aggregate-limit <i>aggregate-packet</i>	(Optional) Sets the total number of buffered packets before some packets may be dropped. Below this limit, packets will not be dropped.
Step 4	Router(config-if)# fair-queue individual-limit <i>individual-packet</i>	(Optional) Sets the maximum queue size for every per-flow queue during periods of congestion.
Step 5	Router(config-if)# fair-queue tos number limit <i>class-packet</i>	(Optional) Sets the maximum queue size for a specific ToS queue during periods of congestion.

In general, you should not change the aggregate, individual, or class limit value from the default. Use the **fair-queue aggregate-limit**, **fair-queue individual-limit**, and **fair-queue limit** commands only if you have determined that you would benefit from using different values, based on your particular situation.

Monitoring DWFQ

To monitor DWFQ, use one or more of the following commands in EXEC mode:

Command	Purpose
Router# show interfaces [<i>interface</i>]	Displays statistical information specific to an interface.
Router# show queueing fair-queue	Displays status of the fair queueing configuration.

Class-Based Weighted Fair Queueing Configuration Task List

To configure CBWFQ, perform the tasks in the following sections. The first three sections are required; the remaining sections are optional.

- Defining Class Maps (Required)
- Configuring Class Policy in the Policy Map (Required)
- Attaching the Service Policy and Enabling CBWFQ (Required)
- Modifying the Bandwidth for an Existing Policy Map Class (Optional)
- Modifying the Queue Limit for an Existing Policy Map Class (Optional)
- Configuring the Bandwidth Limiting Factor
- Deleting Classes (Optional)
- Deleting Policy Maps (Optional)

- Verifying Configuration of Policy Maps and Their Classes (Optional)

CBWFQ is supported on VBR and ABR ATM connections. It is not supported on UBR connections. See the end of this chapter for the section “CBWFQ Configuration Examples.”

**Note**

For information on how to configure per-VC WFQ and CBWFQ, see the chapter “Configuring IP to ATM Class of Service” in this book.

Defining Class Maps

To create a class map containing match criteria against which a packet is checked to determine if it belongs to a class—and to effectively create the class whose policy can be specified in one or more policy maps—use the first command in global configuration mode to specify the class map name, then one of the following commands in class map configuration mode:

	Command	Purpose
Step 1	Router(config)# class-map <i>class-map-name</i>	Specifies the name of the class map to be created.
Step 2	Router(config-cmap)# match access-group { <i>access-group</i> name <i>access-group-name</i> }	Specifies the name of the ACL against whose contents packets are checked to determine if they belong to the class. CBWFQ supports numbered and named ACLs.
	or	
	Router(config-cmap)# match input-interface <i>interface-name</i>	Specifies the name of the input interface used as a match criterion against which packets are checked to determine if they belong to the class.
	or	
	Router(config-cmap)# match protocol <i>protocol</i>	Specifies the name of the protocol used as a match criterion against which packets are checked to determine if they belong to the class.
	or	
	Router(config-cmap)# match mpls experimental <i>number</i>	Specifies the value of the EXP field to be used as a match criterion against which packets are checked to determine if they belong to the class.

Configuring Class Policy in the Policy Map

To configure a policy map and create class policies that make up the service policy, use the **policy-map** command to specify the policy map name, then use one or more of the following commands to configure policy for a standard class or the default class:

- **class**
- **bandwidth**
- **fair-queue** (for class-default class only)
- **queue-limit** or **random-detect**

For each class that you define, you can use one or more of the listed commands to configure class policy. For example, you might specify bandwidth for one class and both bandwidth and queue limit for another class.

The default class of the policy map (commonly known as the class-default class) is the class to which traffic is directed if that traffic does not satisfy the match criteria of other classes whose policy is defined in the policy map.

You can configure class policies for as many classes as are defined on the router, up to the maximum of 64. However, the total amount of bandwidth allocated for all classes included in a policy map must not exceed 75 percent of the available bandwidth on the interface. The other 25 percent is used for control and routing traffic. (To override the 75 percent limitation, use the **max-reserved bandwidth** command.) If not all of the bandwidth is allocated, the remaining bandwidth is proportionally allocated among the classes, based on their configured bandwidth.

To configure class policies in a policy map, perform the tasks in the following sections:

- Configuring Class Policy Using Tail Drop
- Configuring Class Policy Using WRED Packet Drop
- Configuring the Class-Default Class Policy

Configuring Class Policy Using Tail Drop

To configure a policy map and create class policies that make up the service policy, use the first command in global configuration mode to specify the policy map name, then use the following commands in policy-map class configuration mode to configure policy for a standard class. To configure policy for the default class, see the section “Configuring the Class-Default Class Policy.”

	Command	Purpose
Step 1	Router(config)# policy-map <i>policy-map</i>	Specifies the name of the policy map to be created or modified.
Step 2	Router(config-pmap)# class <i>class-name</i>	Specifies the name of a class to be created and included in the service policy.
Step 3	Router(config-pmap-c)# bandwidth { <i>bandwidth-kbps</i> percent <i>percent</i> }	Specifies the amount of bandwidth in kbps, or percentage of available bandwidth, to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.
Step 4	Router(config-pmap-c)# queue-limit <i>number-of-packets</i>	Specifies the maximum number of packets that can be enqueued for the class.

To configure policy for more than one class in the same policy map, repeat Step 2 through Step 4. Note that because this set of commands uses the **queue-limit** command, the policy map uses tail drop, not Weighted Random Early Detection (WRED) packet drop.

Configuring Class Policy Using WRED Packet Drop

To configure a policy map and create class policies comprising the service policy, use the first global configuration command to specify the policy map name, then use the following policy-map class configuration commands to configure policy for a standard class. To configure policy for the default class, see the section “Configuring the Class-Default Class Policy.”

	Command	Purpose
Step 1	Router(config)# policy-map <i>policy-map</i>	Specifies the name of the policy map to be created or modified.
Step 2	Router(config-pmap)# class <i>class-name</i>	Specifies the name of a class to be created and included in the service policy.
Step 3	Router(config-pmap-c)# bandwidth { <i>bandwidth-kbps</i> percent <i>percent</i> }	Specifies the amount of bandwidth in kbps, or percentage of available bandwidth, to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.
Step 4	Router(config-pmap-c)# random-detect	Enables WRED. The class policy will drop packets using WRED instead of tail drop.
Step 5	Router(config-pmap-c)# random-detect exponential-weighting-constant <i>exponent</i> and/or Router(config-pmap-c)# random-detect precedence <i>precedence min-threshold max-threshold mark-prob-denominator</i>	Configures the exponential weight factor used in calculating the average queue length. Configures WRED parameters for packets with a specific IP Precedence. Repeat this command for each precedence.

To configure policy for more than one class in the same policy map, repeat Step 2 through Step 5. Note that this set of commands uses WRED packet drop, not tail drop.



Note

If you configure a class in a policy map to use WRED for packet drop instead of tail drop, you must ensure that WRED is not configured on the interface to which you intend to attach that service policy.

Configuring the Class-Default Class Policy

The class-default class is used to classify traffic that does not fall into one of the defined classes. Once a packet is classified, all of the standard mechanisms that can be used to differentiate service among the classes apply. The class-default class was predefined when you created the policy map, but you must configure it. If no default class is configured, then by default the traffic that does not match any of the configured classes is flow classified and given best-effort treatment.

By default, the class-default class is defined as flow-based WFQ. However, configuring the default class with the **bandwidth** policy-map class configuration command disqualifies the default class as flow-based WFQ.

To configure a policy map and configure the class-default class to use tail drop, use the first global configuration command to specify the policy map name, then use the following policy-map class configuration commands to configure policy for the default class:

	Command	Purpose
Step 1	Router(config)# policy-map <i>policy-map</i>	Specifies the name of the policy map to be created or modified.
Step 2	Router(config-pmap)# class class-default <i>default-class-name</i>	Specifies the default class so that you can configure or modify its policy.
Step 3	Router(config-pmap-c)# bandwidth { <i>bandwidth-kbps</i> percent <i>percent</i> } or Router(config-pmap-c)# fair-queue <i>[number-of-dynamic-queues]</i>	Specifies the amount of bandwidth in kbps, or percentage of available bandwidth, to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead. Specifies the number of dynamic queues to be reserved for use by flow-based WFQ running on the default class. The number of dynamic queues is derived from the bandwidth of the interface. See the tables accompanying the description of the fair-queue (WFQ) command in the <i>Cisco IOS Quality of Service Solutions Command Reference</i> for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface or ATM VC.
Step 4	Router(config-pmap-c)# queue-limit <i>number-of-packets</i>	Specifies the maximum number of packets that the queue for the default class can accumulate.

To configure a policy map and configure the class-default class to use WRED packet drop, use the first global configuration command to specify the policy map name, then use the following policy-map class configuration commands to configure policy for the default class:

	Command	Purpose
Step 1	Router(config)# policy-map <i>policy-map</i>	Specifies the name of the policy map to be created or modified.
Step 2	Router(config-pmap)# class class-default <i>default-class-name</i>	Specifies the default class so that you can configure or modify its policy.

	Command	Purpose
Step 3	<pre>Router(config-pmap-c) # bandwidth {<i>bandwidth-kbps</i> percent <i>percent</i>} or Router(config-pmap-c) # fair-queue [<i>number-of-dynamic-queues</i>]</pre>	<p>Specifies the amount of bandwidth in kbps, or percentage of available bandwidth, to be assigned to the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.</p> <p>Specifies the number of dynamic queues to be reserved for use by flow-based WFQ running on the default class. The number of dynamic queues is derived from the bandwidth of the interface. See the tables accompanying the description of the fair-queue (WFQ) command in the <i>Cisco IOS Quality of Service Solutions Command Reference</i> for the default number of dynamic queues that WFQ and CBWFQ use when they are enabled on an interface or ATM VC.</p>
Step 4	<pre>Router(config-pmap-c) # random-detect</pre>	<p>Enables WRED. The class policy will drop packets using WRED instead of tail drop.</p>
Step 5	<pre>Router(config-pmap-c) # random-detect exponential-weighting-constant <i>exponent</i> and/or Router(config-pmap-c) # random-detect precedence <i>precedence min-threshold max-threshold</i> <i>mark-prob-denominator</i></pre>	<p>Configures the exponential weight factor used in calculating the average queue length.</p> <p>Configures WRED parameters for packets with a specific IP Precedence. Repeat this command for each precedence.</p>

Attaching the Service Policy and Enabling CBWFQ

To attach a service policy to the output interface and enable CBWFQ on the interface, use the following interface configuration command. When CBWFQ is enabled, all classes configured as part of the service policy map are installed in the fair queuing system.

Command	Purpose
<pre>Router(config-if) # service-policy output <i>policy-map</i></pre>	<p>Enables CBWFQ and attaches the specified service policy map to the output interface.</p>

Configuring CBWFQ on a physical interface is only possible if the interface is in the default queuing mode. Serial interfaces at E1 (2.048 Mbps) and below use WFQ by default—other interfaces use FIFO by default. Enabling CBWFQ on a physical interface overrides the default interface queuing method. Enabling CBWFQ on an ATM PVC does not override the default queuing method.

Modifying the Bandwidth for an Existing Policy Map Class

To change the amount of bandwidth allocated for an existing class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	<code>Router(config)# policy-map <i>policy-map</i></code>	Specifies the name of the policy map containing the class to be modified.
Step 2	<code>Router(config-pmap)# class <i>class-name</i></code>	Specifies the name of a class whose bandwidth you want to modify.
Step 3	<code>Router(config-pmap-c)# bandwidth {<i>bandwidth-kbps</i> percent <i>percent</i>}</code>	Specifies the new amount of bandwidth in kbps, or percentage of available bandwidth, to be used to reconfigure the class. The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.

Modifying the Queue Limit for an Existing Policy Map Class

To change the maximum number of packets that can accrue in a queue reserved for an existing class, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	<code>Router(config)# policy-map <i>policy-map</i></code>	Specifies the name of the policy map containing the class to be modified.
Step 2	<code>Router(config-pmap)# class <i>class-name</i></code>	Specifies the name of a class whose queue limit you want to modify.
Step 3	<code>Router(config-pmap-c)# queue-limit <i>number-of-packets</i></code>	Specifies the new maximum number of packets that can be enqueued for the class to be reconfigured. The default and maximum number of packets is 64.

Configuring the Bandwidth Limiting Factor

To change the maximum reserved bandwidth allocated for CBWFQ, LLQ, and the IP RTP Priority feature, use the following command in interface configuration mode:

Command	Purpose
<code>Router(config-if)# max-reserved-bandwidth <i>percent</i></code>	Changes the maximum configurable bandwidth for CBWFQ, LLQ, and IP RTP Priority. The default is 75 percent.

Deleting Classes

To delete one or more class maps from a service policy map, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	Router(config)# policy-map <i>policy-map</i>	Specifies the name of the policy map containing the classes to be deleted.
Step 2	Router(config-pmap)# no class <i>class-name</i>	Specifies the name of the class(es) to be deleted.
Step 3	Router(config-pmap-c)# no class class-default	Deletes the default class.

Deleting Policy Maps

To delete a policy map, use the following command in global configuration mode:

Command	Purpose
Router(config)# no policy-map <i>policy-map</i>	Specifies the name of the policy map to be deleted.

Verifying Configuration of Policy Maps and Their Classes

To display the contents of a specific policy map, a specific class from a specific policy map, or all policy maps configured on an interface, use one of the following commands in global configuration mode:

Command	Purpose
Router# show policy-map <i>policy-map</i>	Displays the configuration of all classes that make up the specified policy map.
Router# show policy-map <i>policy-map</i> class <i>class-name</i>	Displays the configuration of the specified class of the specified policy map.
Router# show policy-map interface <i>interface-name</i>	Displays the configuration of all classes configured for all policy maps on the specified interface.
Router# show queue <i>interface-type</i> <i>interface-number</i>	Displays queuing configuration and statistics for a particular interface.

The counters displayed after issuing the **show policy-map interface** command are updated only if congestion is present on the interface.

IP RTP Priority Configuration Task List

To configure IP RTP Priority, perform the tasks in the following sections. The first task is required; the remaining tasks are optional.

- Configuring IP RTP Priority (Required)
- Configuring the Bandwidth Limiting Factor (Optional)
- Verifying IP RTP Priority (Optional)

See the end of this chapter for the section “IP RTP Priority Configuration Examples.”

Configuring IP RTP Priority

To reserve a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports, use the following command in interface configuration mode:

Command	Purpose
Router(config-if)# ip rtp priority <i>starting-rtp-port-number port-number-range bandwidth</i>	Reserves a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports.



Note Because the **ip rtp priority** command gives absolute priority over other traffic, it should be used with care. In the event of congestion, if the traffic exceeds the configured bandwidth, then all the excess traffic is dropped.

The **ip rtp reserve** and **ip rtp priority** commands cannot be configured on the same interface.



Note The **frame-relay ip rtp priority** command provides strict priority queueing for Frame Relay PVCs. For more information, refer to the *Cisco IOS Quality of Service Solutions Command Reference*.

Configuring the Bandwidth Limiting Factor

To change the maximum reserved bandwidth allocated for CBWFQ, LLQ, and the IP RTP Priority feature, use the following command in interface configuration mode:

Command	Purpose
Router(config-if)# max-reserved-bandwidth <i>percent</i>	Changes the maximum configurable bandwidth for CBWFQ, LLQ, and IP RTP Priority. The default is 75 percent.

Verifying IP RTP Priority

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# show queue <i>interface-type interface-number</i>	Displays queueing configuration and statistics for a particular interface.

Monitoring and Maintaining IP RTP Priority

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

Command	Purpose
Router# debug priority	Displays priority queueing output if packets are dropped from the priority queue.
Router# show queue <i>interface-type interface-number</i>	Displays queueing configuration and statistics for a particular interface.

Frame Relay IP RTP Priority Configuration Task List

To configure Frame Relay IP RTP Priority, perform the tasks in the following sections. The first task is required; the remaining tasks are optional.

- Configuring IP RTP Priority (Required)
- Verifying IP RTP Priority (Optional)
- Monitoring and Maintaining Frame Relay IP RTP Priority (Optional)

See the end of this chapter for the section “Frame Relay IP RTP Priority Configuration Examples.”

Configuring Frame Relay IP RTP Priority

To reserve a strict priority queue on a Frame Relay PVC for a set of RTP packet flows belonging to a range of UDP destination ports, use the following command in map-class configuration mode:

Command	Purpose
Router(config-map-class)# frame-relay ip rtp priority <i>starting-rtp-port-number port-number-range bandwidth</i>	Reserves a strict priority queue for a set of RTP packet flows belonging to a range of UDP destination ports.

**Note**

Because the **frame-relay ip rtp priority** command gives absolute priority over other traffic, it should be used with care. In the event of congestion, if the traffic exceeds the configured bandwidth, then all the excess traffic is dropped.

Verifying Frame Relay IP RTP Priority

To verify the Frame Relay IP RTP Priority feature, use one of the following commands in EXEC mode:

Command	Purpose
Router# show frame relay pvc	Displays statistics about PVCs for Frame Relay interfaces.
Router# show queue <i>interface-type interface-number</i>	Displays fair queueing configuration and statistics for a particular interface.
Router# show traffic-shape queue	Displays information about the elements queued at a particular time at the VC data link connection identifier (DLCI) level.

Monitoring and Maintaining Frame Relay IP RTP Priority

To tune your RTP bandwidth or decrease RTP traffic if the priority queue is experiencing drops, use the following command in EXEC mode:

Command	Purpose
Router# debug priority	Displays priority queueing output if packets are dropped from the priority queue.

Low Latency Queueing Configuration Task List

To configure low latency queueing, perform the tasks in the following sections. The first section is required; the remaining sections are optional.

- Configuring Low Latency Queueing (Required)
- Configuring the Bandwidth Limiting Factor (Optional)
- Verifying Low Latency Queueing (Optional)

See the end of this chapter for the section “Low Latency Queueing Configuration Examples.”

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)# priority <i>bandwidth</i>	Reserves a strict priority queue for this class of traffic.

Configuring the Bandwidth Limiting Factor

To change the maximum reserved bandwidth allocated for CBWFQ, LLQ, and the IP RTP Priority feature, use the following command in interface configuration mode:

Command	Purpose
Router(config-if)# max-reserved-bandwidth <i>percent</i>	Changes the maximum configurable bandwidth for CBWFQ, LLQ, and IP RTP Priority. The default is 75 percent.

Verifying Low Latency Queuing

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# show queue <i>interface-type interface-number</i>	Displays queuing 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 Queuing

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

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

Flow-Based WFQ Configuration Examples

The following example requests a fair queue with a congestive discard threshold of 64 messages, 512 dynamic queues, and 18 RSVP queues:

```
Router(config)# interface Serial 3/0
Router(config-if)# ip unnumbered Ethernet 0/0
Router(config-if)# fair-queue 64 512 18
```

For information on how to configure WFQ, see the section “Flow-Based Weighted Fair Queuing Configuration Task List” earlier in this chapter.

DWFQ Configuration Examples

The following examples provide DWFQ configuration examples:

- Flow-Based DWFQ Example
- QoS-Group-Based DWFQ Example
- ToS-Based DWFQ Example

For information on how to configure DWFQ, see the section “VIP-Distributed Weighted Fair Queuing Configuration Task List” earlier in this chapter.

Flow-Based DWFQ Example

The following example enables DWFQ on the HSSI 0/0/0 interface:

```
Router(config)# interface Hssi0/0/0
Router(config-if)# description 45Mbps to R2
Router(config-if)# ip address 200.200.14.250 255.255.255.252
Router(config-if)# fair-queue
```

The following is sample output from the **show interfaces fair-queue** command for this configuration:

```
Router# show interfaces hssi 0/0/0 fair-queue

Hssi0/0/0 queue size 0
      packets output 35, drops 0
WFQ: global queue limit 401, local queue limit 200
```

QoS-Group-Based DWFQ Example

The following example configures QoS-group-based DWFQ. CAR policies are used to assign packets with an IP precedence of 2 to QoS group 2, and packets with IP precedence 6 are assigned to QoS group 6.

```
Router(config)# interface Hssi0/0/0
Router(config-if)# ip address 188.1.3.70 255.255.255.0
Router(config-if)# rate-limit output access-group rate-limit 6 155000000 2000000 8000000
conform-action set-qos-transmit 6 exceed-action drop
Router(config-if)# rate-limit output access-group rate-limit 2 155000000 2000000 8000000
conform-action set-qos-transmit 2 exceed-action drop
Router(config-if)# fair-queue qos-group
Router(config-if)# fair-queue qos-group 2 weight 10
Router(config-if)# fair-queue qos-group 2 limit 27
Router(config-if)# fair-queue qos-group 6 weight 30
Router(config-if)# fair-queue qos-group 6 limit 27
!
Router(config)# access-list rate-limit 2 2
Router(config)# access-list rate-limit 6 6
```

Use the **show interfaces fair-queue** command to view WFQ statistics.

```
Router# show interfaces fair-queue

Hssi0/0/0 queue size 0
      packets output 806232, drops 1
WFQ: aggregate queue limit 54, individual queue limit 27
      max available buffers 54

      Class 0: weight 60 limit 27 qsize 0 packets output 654 drops 0
      Class 2: weight 10 limit 27 qsize 0 packets output 402789 drops 0
      Class 6: weight 30 limit 27 qsize 0 packets output 402789 drops 1
```

ToS-Based DWFQ Example

The following example configures ToS-based DWFQ using the default parameters:

```
Router# configure terminal
Router(config)# interface Hssi0/0/0
Router(config-if)# fair-queue tos
Router(config-if)# end
```

The following is output of the **show running-config** command for the Hssi0/0/0 interface. Notice that the router automatically adds the default weights and limits for the ToS classes to the configuration.

```
interface Hssi0/0/0
 ip address 188.1.3.70 255.255.255.0
 fair-queue tos
 fair-queue tos 1 weight 20
 fair-queue tos 1 limit 27
 fair-queue tos 2 weight 30
 fair-queue tos 2 limit 27
 fair-queue tos 3 weight 40
 fair-queue tos 3 limit 27
```

Use the **show interfaces fair-queue** command to view DWFQ statistics.

```
Router# show interfaces fair-queue

Hssi0/0/0 queue size 0
      packets output 1417079, drops 2
WFQ: aggregate queue limit 54, individual queue limit 27
      max available buffers 54

      Class 0: weight 10 limit 27 qsize 0 packets output 1150 drops 0
      Class 1: weight 20 limit 27 qsize 0 packets output 0 drops 0
      Class 2: weight 30 limit 27 qsize 0 packets output 775482 drops 1
      Class 3: weight 40 limit 27 qsize 0 packets output 0 drops 0
```

CBWFQ Configuration Examples

The following sections provide CBWFQ configuration examples:

- Class Map Configuration Example
- Policy Creation Example
- Policy Attachment to Interfaces Example
- CBWFQ Using WRED Packet Drop Example
- Display Service Policy Map Content Examples

For information on how to configure CBWFQ, see the section “Class-Based Weighted Fair Queuing Configuration Task List” earlier in this chapter.

Class Map Configuration Example

In the following example, ACLs 101 and 102 are created. Next, two class maps are created and their match criteria are defined. For the first map class, called class1, the numbered ACL 101 is used as the match criterion. For the second map class, called class2, the numbered ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

```
Router(config)# access-list 101 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

Router(config)# class-map class1
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit

Router(config-cmap)# class-map class2
Router(config-cmap)# match access-group 102
Router(config-cmap)# exit
```

Policy Creation Example

In the following example, a policy map called `policy1` is defined to contain policy specification for the two classes—`class1` and `class2`. The match criteria for these classes were defined in the previous section “Class Map Configuration Example.”

For `class1`, the policy specifies the bandwidth allocation request and the maximum number of packets that the queue for this class can accumulate. For `class2`, the policy specifies only the bandwidth allocation request, so the default queue limit of 64 packets is assumed.

```
Router(config)# policy-map policy1

Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 3000
Router(config-pmap-c)# queue-limit 30
Router(config-pmap-c)# exit

Router(config-pmap)# class class2
Router(config-pmap-c)# bandwidth 2000
Router(config-pmap-c)# exit
```

Policy Attachment to Interfaces Example

The following example shows how to attach an existing policy map. After you define a policy map, you can attach it to one or more interfaces to specify the service policy for those interfaces. Although you can assign the same policy map to multiple interfaces, each interface can have only one policy map attached at the input and one policy map attached at the output.

The policy map in this example was defined in the previous section, “Policy Creation Example.”

```
Router(config)# interface e1/1
Router(config-if)# service output policy1
Router(config-if)# exit

Router(config)# interface fa1/0/0
Router(config-if)# service output policy1
Router(config-if)# exit
```

CBWFQ Using WRED Packet Drop Example

In the following example, the class map `class1` is created and defined to use the input interface `FastEthernet0/1` as a match criterion to determine if packets belong to the class. Next, the policy map `policy1` is defined to contain policy specification for `class1`, which is configured for WRED packet drop.

```
Router(config)# class-map class1
Router(config-cmap)# match input-interface FastEthernet0/1
!
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 1000
Router(config-pmap-c)# random-detect
!
Router(config)# interface serial10/0
Router(config-if)# service-policy output policy1
!
```

Display Service Policy Map Content Examples

The following examples show how to display the contents of service policy maps. There are four methods to display the contents:

- Display all classes that make up a specified service policy map
- Display all classes configured for all service policy maps
- Display a specified class of a service policy map
- Display all classes configured for all service policy maps on a specified interface

All Classes for a Specified Service Policy Map

The following example displays the contents of the po1 service policy map:

```
Router# show policy-map po1
Policy Map po1
  Weighted Fair Queueing
    Class class1
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class2
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class3
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class4
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class5
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class6
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class7
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class8
      Bandwidth 937 (kbps) Max thresh 64 (packets)
```

All Classes for All Service Policy Maps

The following example displays the contents of all policy maps on the router:

```
Router# show policy-map
Policy Map poH1
  Weighted Fair Queueing
    Class class1
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class2
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class3
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class4
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class5
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class6
      Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class7
```

```

        Bandwidth 937 (kbps) Max thresh 64 (packets)
    Class class8
        Bandwidth 937 (kbps) Max thresh 64 (packets)
Policy Map policy2
  Weighted Fair Queueing
    Class class1
        Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class2
        Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class3
        Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class4
        Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class5
        Bandwidth 300 (kbps) Max thresh 64 (packets)
    Class class6
        Bandwidth 300 (kbps) Max thresh 64 (packets)

```

Specified Class for a Service Policy Map

The following example displays configurations for the class called class7 that belongs to the policy map pol:

```

Router# show policy-map pol class class7
Class class7
  Bandwidth 937 (kbps) Max Thresh 64 (packets)

```

All Classes for All Service Policy Maps on a Specified Interface

The following example displays configurations for classes on the output interface e1/1:

```

Router# show policy-map interface e1/1
Ethernet1/1 output : pol
  Weighted Fair Queueing
    Class class1
      Output Queue: Conversation 264
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11548/0/0
    Class class2
      Output Queue: Conversation 265
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11546/0/0
    Class class3
      Output Queue: Conversation 266
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11546/0/0
    Class class4
      Output Queue: Conversation 267
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11702/0/0
    Class class5
      Output Queue: Conversation 268
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11701/0/0
    Class class6
      Output Queue: Conversation 269
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11702/0/0
    Class class7
      Output Queue: Conversation 270
      Bandwidth 937 (kbps) Max Threshold 64 (packets)
      (total/discards/tail drops) 11857/0/0
    Class class8

```

```
Output Queue: Conversation 271
Bandwidth 937 (kbps) Max Threshold 64 (packets)
(total/discards/tail drops) 11858/1/0
```

IP RTP Priority Configuration Examples

The following sections provide IP RTP Priority configuration examples:

- CBWFQ Configuration Example
- Virtual Template Configuration Example
- Multilink Bundle Configuration Example
- Debug Example

CBWFQ Configuration Example

The following example first defines a CBWFQ configuration and then reserves a strict priority queue:

```
! The following commands define a class map:
router(config)# class-map class1
router(config-cmap)# match access-group 101
router(config-cmap)# exit

! The following commands create and attach a policy map:
router(config)# policy-map policy1
router(config-pmap)# class class1
router(config-pmap-c)# bandwidth 3000
router(config-pmap-c)# queue-limit 30
router(config-pmap-c)# random-detect
router(config-pmap-c)# random-detect precedence 0 32 256 100
router(config-pmap-c)# exit
router(config)# interface Serial1
router(config-if)# service-policy output policy1

! The following command reserves a strict priority queue:
router(config-if)# ip rtp priority 16384 16383 40
```

The **queue-limit** and **random-detect** commands are optional commands for CBWFQ configurations. The **queue-limit** command is used for configuring tail drop limits for a class queue. The **random-detect** command is used for configuring RED drop limits for a class queue, similar to the **random-detect** command available on an interface.

Virtual Template Configuration Example

The following example configures a strict priority queue in a virtual template configuration with CBWFQ. The **max-reserved-bandwidth** command changes the maximum reserved bandwidth allocated for CBWFQ and IP RTP Priority from the default (75 percent) to 80 percent.

```
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)# ip rtp priority 16384 16383 25
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)# max-reserved-bandwidth 80
router(config-if)# end

router(config)# interface Serial0/1
router(config-if)# bandwidth 64
router(config-if)# ip address 1.1.1.2 255.255.255.0
router(config-if)# no ip directed-broadcast
router(config-if)# encapsulation ppp
router(config-if)# ppp multilink
router(config-if)# end
```



Note

To make the virtual-access interface function properly, the **bandwidth** policy-map class configuration command should not be configured on the virtual template. It needs to be configured on the actual interface, as shown in the example.

Multilink Bundle Configuration Example

The following example configures a strict priority queue in a multilink bundle configuration with WFQ. The advantage to using multilink bundles is that you can specify different **ip rtp priority** parameters on different interfaces.

The following commands create multilink bundle 1, which is configured for a maximum **ip rtp priority** bandwidth of 200 kbps. The **max-reserved-bandwidth** command changes the maximum reserved bandwidth allocated for WFQ and IP RTP Priority.

```
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)# ip rtp priority 16384 16383 200
router(config-if)# no ip mroute-cache
router(config-if)# fair-queue 64 256 0
router(config-if)# ppp multilink
router(config-if)# ppp multilink fragment-delay 20
router(config-if)# ppp multilink interleave
router(config-if)# max-reserved-bandwidth 80
```

The following commands create multilink bundle 2, which is configured for a maximum **ip rtp priority** bandwidth of 100 kbps:

```
router(config)# interface multilink 2
router(config-if)# ip address 172.17.254.162 255.255.255.248
router(config-if)# no ip directed-broadcast
router(config-if)# ip rtp priority 16384 16383 100
router(config-if)# no ip mroute-cache
router(config-if)# fair-queue 64 256 0
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:

```
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 ip mroute-cache
router(config-if)# no fair-queue
router(config-if)# clockrate 256000
router(config-if)# ppp multilink
router(config-if)# multilink-group 1
```

Next, serial interface 2/1 is configured to be part of multilink bundle 2:

```
router(config)# interface serial 2/1
router(config-if)# bandwidth 128
router(config-if)# no ip address
router(config-if)# no ip directed-broadcast
router(config-if)# encapsulation ppp
router(config-if)# no ip mroute-cache
router(config-if)# no fair-queue
router(config-if)# clockrate 128000
router(config-if)# ppp multilink
router(config-if)# multilink-group 2
```

Debug Example

The following example shows a sample output from the **debug priority** command:

```
Router# debug priority
*Feb 28 16:46:05.659:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.671:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.679:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.691:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.699:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.711:WFQ:dropping a packet from the priority queue 64
*Feb 28 16:46:05.719:WFQ:dropping a packet from the priority queue 64
```

In this example, 64 indicates the actual priority queue depth at the time the packet was dropped.

Frame Relay IP RTP Priority Configuration Examples

This section provides the following configuration examples:

This section provides the following configuration examples:

- Strict Priority Service to Matching RTP Packets Example

For information on how to configure Frame Relay IP RTP Priority queueing, see the section “Frame Relay IP RTP Priority Configuration Task List” earlier in this chapter.

Strict Priority Service to Matching RTP Packets Example

The following example first configures the Frame Relay map class called voip and then applies the map class to PVC 100 to provide strict priority service to matching RTP packets:

```
map-class frame-relay voip
  frame-relay cir 256000
  frame-relay bc 2560
  frame-relay be 600
  frame-relay mincir 256000
  no frame-relay adaptive-shaping
  frame-relay fair-queue
  frame-relay fragment 250
  frame-relay ip rtp priority 16384 16380 210

interface Serial5/0
  ip address 10.10.10.10 255.0.0.0
  no ip directed-broadcast
  encapsulation frame-relay
  no ip mroute-cache
  load-interval 30
  clockrate 1007616
  frame-relay traffic-shaping
  frame-relay interface-dlci 100
    class voip
  frame-relay ip rtp header-compression
  frame-relay intf-type dce
```

In this example, RTP packets on PVC 100 with UDP ports in the range 16384 to 32764 will be matched and given strict priority service.

Low Latency Queueing Configuration Examples

The following sections provide low latency queueing configuration examples:

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

For information on how to configure low latency queueing, see the section “Frame Relay IP RTP Priority Configuration Task List” earlier in this chapter.

ATM PVC Configuration Example

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 the PVC interface 0/102 on the subinterface atm1/0.2.

```
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 atm1/0.2
router(config-subif)# pvc 0/102
router(config-subif-vc)# service-policy output policy1
```

Virtual Template Configuration Example

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 Example

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
```

