

Managing System Performance

This chapter describes the basic tasks that you can perform to manage the general system performance.

For a complete description of the performance management commands in this chapter, refer to the “Performance Management Commands” chapter of the *Configuration Fundamentals Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

System Performance Management Task List

Perform any of the tasks in the following sections to manage system performance:

- Set the Interval for Load Data
- Limit TCP Transactions
- Configure Switching and Scheduling Priorities
- Establish Queueing and Congestion Strategies
- Configure Generic Traffic Shaping
- Modify the System Buffer Size

In addition, most chapters in this guide include performance tasks specific to the chapter content, and the *Internetworking Design Guide* includes detailed information on performance issues that arise when designing a network.

See the “Performance Management Examples” section at the end of this chapter for examples.

Set the Interval for Load Data

You can change the period of time over which a set of data is used for computing load statistics. Decisions, such as dial backup decisions, are dependent on these statistics. If you decrease the load interval, the average statistics are computed over a shorter period of time and are more responsive to bursts of traffic.

To change the length of time for which a set of data is used to compute load statistics, perform the following task in interface configuration mode:

Task	Command
Set the length of time for which data is used for load calculations.	load-interval <i>seconds</i>

Limit TCP Transactions

When using a standard TCP implementation to send keystrokes between machines, TCP tends to send one packet for each keystroke typed, which can use up bandwidth and contribute to congestion on larger networks.

John Nagle’s algorithm (RFC 896) helps alleviate the small-packet problem in TCP. The first character typed after connection establishment is sent in a single packet, but TCP holds any additional characters typed until the receiver acknowledges the previous packet. Then the second, larger packet is sent, and additional typed characters are saved until the acknowledgment comes back. The effect is to accumulate characters into larger chunks, and pace them out to the network at a rate matching the round-trip time of the given connection. This method is usually good for all TCP-based traffic. However, do not enable the Nagle slow packet avoidance algorithm if you have XRemote users on X Window sessions.

By default, the Nagle algorithm is not enabled. To enable the Nagle algorithm and thereby reduce TCP transactions, perform the following task in global configuration mode:

Task	Command
Enable the Nagle slow packet avoidance algorithm.	service nagle

Configure Switching and Scheduling Priorities

The normal operation of the network server allows the switching operations to use as much of the central processor as is required. If the network is running unusually heavy loads that do not allow the processor the time to handle the routing protocols, you might need to give priority to the system process scheduler. To do so, perform the following task in global configuration mode:

Task	Command
Define the maximum amount of time that can elapse without running the lowest-priority system processes.	scheduler interval <i>milliseconds</i>

To change the amount of time that the CPU spends on fast switching and process level operations on the Cisco 7200 series and Cisco 7500 series, perform the following task in global configuration mode:

Task	Command
For the Cisco 7200 series and Cisco 7500 series, change the default time the CPU spends on process tasks and fast switching.	scheduler allocate <i>network-microseconds</i> <i>process-microseconds</i>



Caution Cisco recommends that you do not change the default values of the **scheduler allocate** command.

Establish Queueing and Congestion Strategies

There are four possible queueing algorithms used: first-come-first-serve (FCFS), weighted fair queueing, priority queueing, and custom queueing. For serial interfaces at E1 (2.048 Mbps) and below, weighted fair queueing is used by default. When no other queueing strategies are configured, all other interfaces use FCFS by default. There is also one congestion avoidance algorithm available: random early detection.

You can configure the Cisco IOS software to support the following types of queueing and congestion strategies for prioritizing network traffic:

- Weighted Fair Queueing
- Priority Queueing
- Custom Queueing
- Random Early Detection

You can configure weighted fair queueing, priority queueing, custom queueing, or random early detection, but you can assign only one type to an interface.

Note Weighted fair queueing, priority queueing, and custom queueing are not supported on tunnels.

Weighted Fair Queueing

When enabled for an interface, weighted fair queueing provides traffic priority management that automatically sorts among individual traffic streams without requiring that you first define access lists.

Weighted fair queueing can manage duplex data streams, such as those between pairs of applications, and simplex data streams such as voice or video. From the perspective of weighted fair queueing, there are two categories of data streams: 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.

When you enable weighted fair queueing for an interface, new messages for high-bandwidth conversations are discarded after the congestive-messages threshold you set or the default one 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 are specified by the threshold number.

Priority Queueing

Priority output queueing is a mechanism that allows the administrator to set priorities on the type of traffic passing through the network. Packets are classified according to various criteria, including protocol and subprotocol type, and then queued on one of four output queues (high, medium, normal, and low).

When the server is ready to transmit a packet, it scans the priority queues in order, from highest to lowest, to find the highest-priority packet. After that packet is completely transmitted, the server scans the priority queues again. If a priority output queue fills up, packets are dropped and, for IP, quench indications are sent to the original transmitter.

Although you can enable priority output queueing for any interface, the intended application was for low-bandwidth, congested serial interfaces. Cisco's priority output queueing mechanism allows traffic control based on protocol or interface type. You can also set the size of the queue and defaults for what happens to packets that are not defined by priority output queue rules.

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.

Note Priority queueing introduces extra overhead that is acceptable for slow interfaces, but may not be acceptable for higher-speed interfaces such as Ethernet.

The four priority queues—high, medium, normal, and low—are listed in order from highest to lowest priority. 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.

A priority list is a set of rules that describes 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.

Custom Queueing

Priority queueing introduces a fairness problem in that packets classified to lower-priority queues might not get serviced in a timely manner or at all, depending upon the bandwidth used by packets sent from the higher-priority output queues.

With custom output queueing, a “weighted fair” queueing strategy is implemented for the processing of interface output queues. You can control the percentage of an interface's available bandwidth that is used by a particular kind of traffic. When custom queueing is enabled on an interface, the system maintains 17 output queues for that interface that can be used to modify queueing behavior. You can specify queues 1 through 16.

For queue numbers 1 through 16, the system cycles through the queues sequentially, delivering packets in the current queue before moving on to the next. 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. When a particular queue is being processed, packets are sent until the number of bytes sent exceed 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.

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, to this queue. Other traffic cannot be configured to use this queue.

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

Note With custom or priority queueing enabled, the system takes longer to switch packets because the packets are classified by the processor card.

Random Early Detection

Random early detection is useful in high-speed networks to provide a congestion avoidance mechanism (as opposed to a congestion management mechanism such as queueing). When enabled on an interface, random early detection begins dropping packets at a rate you select during configuration when congestion occurs.

Random early detection is recommended only for TCP/IP networks. You can use random early detection as a way to cause TCP to back off traffic. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

Random early detection is not recommended for protocols, such as AppleTalk or Novell Netware, that respond to dropped packets by retransmitting the packets at the same rate. Random early detection should only be configured on an interface where most of the traffic is TCP/IP traffic.

For interfaces configured to use RSVP, random early detection chooses packets from other flows to drop rather than the RSVP flows. Also, IP precedence governs which packets are dropped—traffic that is at a lower precedence has a higher drop rate and therefore is more likely to be throttled back.

Queueing Task List

You can set up weighted fair queueing, priority queueing, custom queueing, or random early detection on your network, but you can assign only one of the four to an interface.

The following sections describe the tasks that you can choose from, depending on the needs of your network:

- Set Weighted Fair Queueing for an Interface
- Enable Priority Queueing
- Enable Custom Queueing
- Enable Random Early Detection on an Interface

Note To configure priority queueing or custom queueing over X.25 or LAPB, refer to the “Configuring X.25 and LAPB” chapter in the *Wide-Area Networking Configuration Guide*.

Set Weighted Fair Queueing for an Interface

To enable weighted fair queueing for an interface, set the congestion threshold after which messages for high-bandwidth conversations are dropped, and specify the number of dynamic and reservable queues, perform the following task in interface configuration mode after specifying the interface:

Task	Command
Configure an interface to use weighted fair queueing.	fair-queue [<i>congestive-discard-threshold</i> [<i>dynamic-queues</i> [<i>reservable-queues</i>]]]

To disable weighted fair queueing for an interface, use the **no fair-queue** command.

Fair queueing is enabled by default for physical interfaces whose bandwidth is less than or equal to 2.048 megabits per second (Mbps) and that do not use Link Access Procedure, Balanced (LAPB), X.25, or Synchronous Data Link Control (SDLC) encapsulations. (Fair queueing is not an option for these protocols.) However, if custom queueing or priority queueing is enabled for a qualifying link,

it overrides fair queueing, effectively disabling it. Additionally, fair queueing is automatically disabled if you enable autonomous or SSE switching. Fair queueing is now enabled by default on interfaces configured for Multilink PPP.

Enable Priority Queueing

To enable priority queueing, perform the tasks in the following sections. The first and third tasks are required.

- Assign Packets to Priority Queues
- Specify the Maximum Packets in the Priority Queues
- Assign a Priority Group to an Interface
- Monitor the Priority Queueing Lists

Assign Packets to Priority Queues

You can assign packets to priority lists based on the protocol type or the interface where the packets enter the router. In addition, you can set the default queue for packets which do not match other assignment rules. To define the priority lists, perform the following tasks in global configuration mode:

Task	Command
Establish queueing priorities based upon the protocol type.	priority-list <i>list-number</i> protocol <i>protocol-name</i> { high medium normal low } <i>queue-keyword</i> <i>keyword-value</i>
Establish queueing priorities for packets entering from a given interface.	priority-list <i>list-number</i> interface <i>interface-type</i> <i>interface-number</i> { high medium normal low }
Assign a priority queue for those packets that do not match any other rule in the priority list.	priority-list <i>list-number</i> default { high medium normal low }

All protocols supported by Cisco are allowed. The *queue-keyword* variable provides additional options including byte-count, TCP service and port number assignments, and AppleTalk, IP, IPX, VINES, or XNS access list assignments. See the **priority-list** command syntax description in the “Performance Management Commands” chapter in the *Configuration Fundamentals Command Reference*.

When using multiple rules, remember that the system reads the **priority-list** commands in order of appearance. When classifying a packet, the system searches the list of rules specified by **priority-list** commands for a matching protocol or interface type. When a match is found, the packet is assigned to the appropriate queue. The list is searched in the order it is specified, and the first matching rule terminates the search.

Specify the Maximum Packets in the Priority Queues

You can specify the maximum number of packets allowed in each of the priority queues. To do so, perform the following task in global configuration mode:

Task	Command
Specify the maximum number of packets allowed in each of the priority queues.	priority-list <i>list-number</i> queue-limit <i>high-limit</i> <i>medium-limit</i> <i>normal-limit</i> <i>low-limit</i>

Assign a Priority Group to an Interface

You can assign a priority list number to an interface. Only one list can be assigned per interface. To assign an priority group to an interface, perform the following task in interface configuration mode:

Task	Command
Assign a priority list number to the interface.	priority-group <i>list-number</i>

Monitor the Priority Queueing Lists

You can display information about the input and output queues when priority queueing is enabled on an interface. To do so, perform the following task in EXEC mode:

Task	Command
Show the status of the priority queueing lists.	show queueing priority

Enable Custom Queueing

To enable custom queueing, perform the tasks in the following sections. The first and third tasks are required.

- Assign Packets to Custom Queues
- Specify the Maximum Packets and Bytes in the Custom Queues
- Assign a Custom Queue to an Interface
- Monitor the Custom Queueing Lists

Assign Packets to Custom Queues

You can assign packets to custom queues based on the protocol type or the interface where the packets enter the router. In addition, you can set the default queue for packets which do not match other assignment rules. To define the custom queueing lists, perform the following tasks in global configuration mode:

Task	Command
Establish queueing priorities based upon the protocol type.	queue-list <i>list-number</i> protocol <i>protocol-name</i> <i>queue-number</i> <i>queue-keyword</i> <i>keyword-value</i>
Establish custom queueing based on packets entering from a given interface.	queue-list <i>list-number</i> interface <i>interface-type</i> <i>interface-number</i> <i>queue-number</i>
Assign a queue number for those packets that do not match any other rule in the custom queue list.	queue-list <i>list-number</i> default <i>queue-number</i>

All protocols supported by Cisco are allowed. The *queue-keyword* variable provides additional options, including byte-count, TCP service and port number assignments, and AppleTalk, IP, IPX, VINES, or XNS access list assignments. See the **queue-list** command syntax description in the “Performance Management Commands” chapter in the *Configuration Fundamentals Command Reference*.

When using multiple rules, remember that the system reads the **queue-list** commands in order of appearance. When classifying a packet, the system searches the list of rules specified by **queue-list** commands for a matching protocol or interface type. When a match is found, the packet is assigned to the appropriate queue. The list is searched in the order it is specified, and the first matching rule terminates the search.

Specify the Maximum Packets and Bytes in the Custom Queues

You can specify the maximum number of packets allowed in each of the custom queues or the maximum queue size in bytes. To do so, perform one of the following tasks in global configuration mode:

Task	Command
Specify the maximum number of packets allowed in each of the custom queues.	queue-list <i>list-number</i> queue <i>queue-number</i> limit <i>limit-number</i>
Designate the byte size allowed per queue.	queue-list <i>list-number</i> queue <i>queue-number</i> byte-count <i>byte-count-number</i>

Assign a Custom Queue to an Interface

You can assign a custom queue list number to an interface. Only one list can be assigned per interface. To assign an custom queue to an interface, perform the following task in interface configuration mode:

Task	Command
Assign a custom queue list number to the interface.	custom-queue-list <i>list-number</i>

Monitor the Custom Queueing Lists

You can display information about the input and output queues when custom queueing is enabled on an interface. To do so, perform one of the following tasks in EXEC mode:

Task	Command
Show the status of the custom queueing lists.	show queueing custom
Show the current status of the custom output queues when custom queueing is enabled.	show interface <i>type number</i>

Enable Random Early Detection on an Interface

To enable random early detection on the interface, perform the following task in interface configuration mode:

Task	Command
Enable random early detection on an interface.	random-detect [<i>weighting</i>]

To monitor the various drop statistics for early random detection, perform the following tasks in EXEC mode:

Task	Command
Show the drop statistics for the interface.	show interface [<i>type number</i>]

Configure Generic Traffic Shaping

Traffic shaping allows you to control how fast packets are sent out on the interface to avoid congestion and meet the needs of remote interfaces. You may want to configure traffic shaping on the interface if you have a network with differing access rates or if you are offering a subrate service. For example, if one end of the link in a Frame Relay network is 256 kbps and the other end of the link is only 128 kbps, sending packets at 256 kbps could cause failure of the applications using the link.

Traffic shaping is supported on all media and encapsulation types on the router. Traffic shaping can also be applied to a specific access list on an interface. To perform traffic shaping on Frame Relay virtual circuits, you can also use the **frame-relay traffic-shaping** command. For more information on Frame Relay traffic shaping, refer to the “Configuring Frame Relay” chapter in the *Wide-Area Network Configuration Guide*.

To enable traffic shaping for outbound traffic on an interface, perform one of the following tasks in interface configuration mode:

Task	Command
Enable traffic shaping for outbound traffic on an interface.	traffic-shape rate <i>bit-rate</i> [<i>burst-size</i> [<i>excess-burst-size</i>]]
Enable traffic shaping for outbound traffic on an interface for a specified access list.	traffic-shape group <i>access-list</i> <i>bit-rate</i> [<i>burst-size</i> [<i>excess-burst-size</i>]]

Note Traffic shaping is not supported with distributed or flow switching. If you enable traffic shaping, all interfaces will revert to fast switching.

If traffic shaping is performed on a Frame Relay network with the **traffic-shape rate** command, you can also use the **traffic-shape adaptive** command to specify the minimum bit rate the traffic is shaped to. Use the **traffic-shape fecn-adapt** to generate BECNs to notify senders to adapt traffic.

To configure a Frame Relay subinterface to estimate the available bandwidth when backward explicit congestion notifications (BECNs) are received, perform the following task in interface configuration mode:

Task	Command
Configure minimum bit rate that traffic is shaped to when BECNs are received on an interface.	traffic-shape adaptive [<i>bit-rate</i>]
Configure reflection of FECN signals as BECNs.	traffic-shape fecn-adapt

The **traffic-shape adaptive** command uses the configured bit rate as a lower bound of the range and the bit rate specified by the **traffic-shape rate** command as the upper bound. The rate that the traffic is actually shaped to can be the upper bound, lower bound, or anywhere between the rates specified with **traffic-shape adaptive** and **traffic-shape rate** commands. Configure the **traffic-shape adaptive** command on the end where traffic needs to be shaped at the lower bound, in case of congestion.

To display the current traffic-shaping configuration and statistics, perform the following tasks in EXEC mode:

Task	Command
Display the current traffic-shaping configuration.	show traffic-shape [<i>interface</i>]
Display the current traffic-shaping statistics.	show traffic-shape statistics [<i>interface</i>]

For an example of configuring traffic shaping, see the section “Generic Traffic Shaping Example” at the end of this chapter.

Modify the System Buffer Size

You can adjust initial buffer pool settings and the limits at which temporary buffers are created and destroyed. To do so, perform the following tasks in global configuration mode:

Task	Command
Adjust the system buffer sizes.	buffers { <i>small</i> <i>middle</i> <i>big</i> <i>verybig</i> <i>large</i> <i>huge</i> <i>type number</i> } { <i>permanent</i> <i>max-free</i> <i>min-free</i> <i>initial</i> } <i>number</i>
Dynamically resize all huge buffers to the value that you supply.	buffers huge size <i>number</i>



Caution Normally you need not adjust these parameters; do so only after consulting with technical support personnel. Improper settings can adversely impact system performance.

During normal system operation, there are two sets of buffer pools: public and interface.

- The buffers in the public pools grow and shrink based upon demand. Some public pools are temporary and are created and destroyed as needed. Other public pools are permanently allocated and cannot be destroyed. The public buffer pools are small, middle, big, large, very big, and huge.
- Interface pools are static—that is, they are all permanent. One interface pool exists for each interface. For example, a Cisco 4000 1E 4T configuration has one Ethernet buffer pool and four serial buffer pools. In the **buffers** command, the *type* and *number* arguments allow the user to tune the interface pools.

See the section “Buffer Modification Examples” at the end of this chapter.

The server has one pool of queueing elements and six public pools of packet buffers of different sizes. For each pool, the server keeps count of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. To display statistics about the buffer pool on the system, perform the following tasks in EXEC mode

Task	Command
Display all public pool information.	show buffers
Display buffer information for an address.	show buffers address <i>hex-addr</i>
Display all public and interface pool information.	show buffers all [<i>dump</i> <i>header</i> <i>packet</i>]
Display a listing of all buffers in use.	show buffers assigned [<i>dump</i> <i>header</i> <i>packet</i>]
Display buffer allocation failures	show buffers failures [<i>dump</i> <i>header</i> <i>packet</i>]
Display buffers available for use	show buffers free [<i>dump</i> <i>header</i> <i>packet</i>]

Task	Command
Display buffers older than one minute.	show buffers old [dump header packet]
Display buffer information for an input interface.	show buffers input-interface <i>interface-type identifier</i>
Display interface pool information.	show buffers pool <i>pool name</i>

Performance Management Examples

The following sections provide performance management examples:

- Generic Traffic Shaping Example
- Buffer Modification Examples

Generic Traffic Shaping Example

This example shows the configuration of two traffic-shaped interfaces on a router. Ethernet 0 is configured to limit User Datagram Protocol (UDP) traffic to 1 Mbps. Ethernet 1 is configured to limit all output to 5 Mbps.

```
access-list 101 permit udp any any
interface Ethernet0
  traffic-shape group 101 1000000 125000 125000
!
interface Ethernet1
  traffic-shape rate 5000000 625000 625000
```

The following is a sample display for the **show traffic-shape** command for the example shown:

```
Router# show traffic-shape
```

I/F	access list	Target Rate	Byte Limit	Sustain bits/int	Excess bits/int	Interval (ms)	Increment (bytes)	Adapt Active
Et0	101	1000000	23437	125000	125000	63	7813	-
Et1		5000000	87889	625000	625000	16	9766	-

The following is a sample display for the **show traffic-shape statistics** command for the example shown:

```
Router# show traffic-shape statistics
```

I/F	Access List	Queue Depth	Packets	Bytes	Packets Delayed	Bytes Delayed	Shaping Active
Et0	101	0	2	180	0	0	no
Et1		0	0	0	0	0	no

Buffer Modification Examples

The following example instructs the system to keep at least 50 small buffers free:

```
buffers small min-free 50
```

The following example instructs the system to keep no more than 200 medium buffers free:

```
buffers middle max-free 200
```

The following example instructs the system to create one large temporary extra buffer, just after a reload:

```
buffers large initial 1
```

The following example instructs the system to create one permanent huge buffer:

```
buffers huge permanent 1
```