Configuring Modular QoS Congestion Avoidance

Congestion avoidance techniques monitor traffic flow in an effort to anticipate and avoid congestion at common network bottlenecks. Avoidance techniques are implemented before congestion occurs as compared with congestion management techniques that control congestion after it has occurred.

Congestion avoidance is achieved through packet dropping. Cisco IOS XR software supports these quality of service (QoS) congestion avoidance techniques that drop packets:

- Random early detection (RED)
- Weighted random early detection (WRED)
- Tail drop

The module describes the concepts and tasks related to these congestion avoidance techniques.

Feature History for Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software

Release	Modification
Release 3.2	The Congestion Avoidance feature was introduced.
Release 3.6.0	Weighted random early detection (WRED) support for IPv4 multicast egress QoS traffic was introduced on 2.5 Gbps IP Services Engine (Engine 3) linecards.
Release 4.0.0	In calculations for the average queue size, indicated that the exponential weight factor is not configurable (CSCeg75763).

Prerequisites for Configuring Modular QoS Congestion Avoidance

This prerequisite is required for configuring QoS congestion avoidance on your network:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Configuring Modular QoS Congestion Avoidance

Random Early Detection and TCP

The Random Early Detection (RED) congestion avoidance technique takes advantage of the congestion control mechanism of TCP. 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 decreases its transmission rate until all packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing traffic bursts. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

Average Queue Size for WRED

The router automatically determines the parameters to use in the WRED calculations. The average queue size is based on the previous average and current size of the queue. The formula is:

average = (old_average * (1-2 -x)) + (current_queue_size * 2 -x)

where *x* is the exponential weight factor.

For high values of *x*, the previous average becomes more important. A large factor smooths out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding a drastic change in size. The WRED process is slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average accommodates temporary bursts in traffic.

Note The exponential weight factor, *x*, is fixed and is not user configurable.

Note If the value of *x* gets too high, WRED does not react to congestion. Packets are sent or dropped as if WRED were not in effect.

For low values of *x*, the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process stops dropping packets.

If the value of *x* gets too low, WRED overreacts to temporary traffic bursts and drops traffic unnecessarily.

WRED Support for IPv4 Multicast Egress QoS Traffic

Weighted random early detection (WRED) is supported for IPv4 multicast egress QoS traffic. The following features are supported:

- Classification based on precedence and differentiated services code point (DSCP)
- Queue selection
- Maximum and minimum guaranteed bandwidth
- Detection of remaining bandwidth
- Priority queuing
- Queue limit
- Random early detection (RED)
- Traffic shaping
- Layer 2 set on ATM and Ethernet

Policy map configurations that use Random Early Detection (RED) must comply with the following requirements. These requirements must be met across all interfaces in the line card for precedence and DSCP-based WRED to be supported for multicast traffic. The policy map configuration requirements are specific to a line card and not across the line cards:

- Multicast must be enabled.
- The same class maps must be used in all the policy maps.
- The class maps must be configured in the same order in all of the policy maps.
- The same RED statements must be configured in a class map across policy maps. The minimum and maximum threshold values can be different for each policy map.
- If a RED profile is used by more than one DSCP or precedence in a class, the precedence or DSCP values share the same RED profile in all the different policy maps.

If the policy map requirements are not met for all interfaces in the line card and are not met for at least one policy map, RED configuration is not applied to the multicast traffic and all multicast traffic in the line card is tail-dropped. See the Tail Drop and the FIFO Queue.

Interfaces with no policy maps and interfaces with policy maps that do not use RED function normally.

For hierarchal policy maps that use WRED, only the child policies should conform to the policy map configuration requirements.

Supported Platforms

WRED is supported for IPv4 multicast egress QoS traffic on the following Cisco XR 12000 Series Router platforms:

- POS line cards
  - Cisco XR 12000 Series 4xOC12c/STM4c POS Rev B
  - Cisco XR 12000 Series 16xOC3c/STM1c POS Rev B
  - Cisco XR 12000 Series 8xOC3c/STM1c POS
  - Cisco XR 12000 Series 4xOC3c/STM1c POS
  - Cisco XR 12000 Series 1xOC48c/STM16c POS
- Ethernet line card
  - Cisco XR 12000 Series 4xGE

Tail Drop and the FIFO Queue

Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. Tail drop treats all traffic flow equally and does not differentiate between classes of service. It manages the packets that are unclassified, placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

See the “Default Traffic Class” section of the “Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software” module.

Configuring Random Early Detection

This configuration task is similar to that used for WRED except that the **random-detect precedence** command is not configured and the **random-detect** command with the **default** keyword must be used to enable RED.

Restrictions

For the **random-detect** command to take effect, you must configure either the **shape average, bandwidth/bandwidth remaining percent** command in the user defined policy map class. This dependency is not applicable to the policy map class class-default.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] }
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value*
6. **shape average** {**percent** *percentage* | *value* [*units*]}
7. **exit**
8. **exit**
9. **interface** *type interface-path-id*
10. **service-policy** {**input** | **output**} *policy-map*
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] } **Example:** `RP/0/0/CPU0:router(config-pmap-c)# random-detect default`	Enables RED with default minimum and maximum thresholds.
Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value* **Example:** `RP/0/0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. or (Optional) Specifies how to allocate leftover bandwidth to various classes.

	Command or Action	Purpose
	or `RP/0/0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	
Step 6	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 7	**exit** **Example:** `RP/0/0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface TenGigE 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 10	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 11	**commit**	

Configuring Weighted Random Early Detection

WRED drops packets selectively based on IP precedence. Edge routers assign IP precedences to packets as they enter the network. WRED uses these precedences to determine how to treat different types of traffic.

When a packet arrives, the following actions occur:

- The average queue size is calculated.
- If the average queue size is less than the minimum queue threshold, the arriving packet is queued.

- If the average queue size is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
- If the average queue size is greater than the maximum threshold, the packet is dropped.

Restrictions

You cannot configure WRED in a class that has been set for priority queueing (PQ).

You cannot use the **random-detect** command in a class configured with the **priority** command.

For the **random-detect** command to take effect, you must configure either the **shape average** or **bandwidth/bandwidth remaining percent** command in the user defined policy map class. This dependency is not applicable to the policy map class class-default.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*]
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]*
6. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
7. **bandwidth remaining percent** *value*
8. **shape average** {**percent** *percentage* | *value* [*units*]}
9. **queue-limit** *value* [*units*]
10. **exit**
11. **interface** *type interface-path-id*
12. **service-policy** {**input** | **output**} *policy-map*
13. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

	Command or Action	Purpose
Step 4	**random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*] **Example:** `RP/0/0/CPU0:router(config-pmap-c)# random-detect dscp af11 1000000 bytes 2000000 bytes`	Modifies the minimum and maximum packet thresholds for the DSCP value. • Enables RED. • *dscp-value*—Number from 0 to 63 that sets the DSCP value. Reserved keywords can be specified instead of numeric values. • *min-threshold*—Minimum threshold in the specified units. The value range of this argument is from 512 to 1073741823. When the average queue length reaches the minimum threshold, WRED randomly drops some packets with the specified DSCP value. • *max-threshold*—Maximum threshold in the specified units. The value range of this argument is from the value of the *min-threshold* argument to 1073741823. When the average queue length exceeds the maximum threshold, WRED drops all packets with the specified DSCP value. • *units*—Units of the threshold value. This can be **bytes**, **gbytes**, **kbytes**, **mbytes**, **ms** (milliseconds), **packets**, or **us** (microseconds). The default is **packets**. • This example shows that for packets with DSCP AF11, the WRED minimum threshold is 1,000,000 bytes and maximum threshold is 2,000,000 bytes.

Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]*
	
	Example:
	
	`RP/0/0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	
	`RP/0/0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`
		(Optional) Specifies the bandwidth allocated for a class belonging to a policy map.
		or
		(Optional) Specifies how to allocate leftover bandwidth to various classes.

Step 6	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class1.
Step 7	**bandwidth remaining percent** *value* **Example:** `RP/0/0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. • The remaining bandwidth of 70 percent is shared by all configured classes. • In this example, class class1 receives 20 percent of the 70 percent.

	Command or Action	Purpose
Step 8	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 9	**queue-limit** *value* [*units*] **Example:** `RP/0/0/CPU0:router(config-pmap-c)# queue-limit 50 ms`	(Optional) Changes queue-limit to fine-tune the amount of buffers available for each queue. The default queue-limit is 100 ms of the service rate for a non-priority class and 10ms of the service rate for a priority class.
Step 10	**exit** **Example:** `RP/0/0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 11	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface pos 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 12	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • In this example, the traffic policy evaluates all traffic leaving that interface. • Ingress policies are not valid; the **bandwidth** and **bandwidth remaining** commands cannot be applied to ingress policies.
Step 13	**commit**	

Configuring Tail Drop

Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are serviced. The **queue-limit** command is used to define the maximum threshold for a class. When the maximum threshold is reached, enqueued packets to the class queue result in tail drop (packet drop).

The **queue-limit** value uses the guaranteed service rate (GSR) of the queue as the reference value for the **queue_bandwidth**. If the class has bandwidth percent associated with it, the **queue-limit** is set to a proportion of the bandwidth reserved for that class.

When a class has no guaranteed service rate, the default queue limit depends on whether shaping is applied. If shaping is not applied, the default queue limit is 16384 packets. If shaping is applied, the default queue limit is:

default queue limit (in packets) = (200 ms * (queue bandwidth or shaper rate) / 8) / average packet size, which is 250 bytes

Restrictions

- When configuring the **queue-limit** command in a class, you must configure one of the following commands: **priority**, **shape average**, **bandwidth**, or **bandwidth remaining**, except for the default class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **queue-limit** *value* [*units*]
5. **priority**[**level** *priority-level*]
6. **police rate percent** *percentage*
7. **class** *class-name*
8. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
9. **bandwidth remaining percent** *value*
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*
13. **service-policy** {**input** | **output**} *policy-map*
14. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and also enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

	Command or Action	Purpose
Step 4	**queue-limit** *value* [*units*] **Example:** `RP/0/0/CPU0:router(config-pmap-c)# queue-limit 1000000 bytes`	Specifies or modifies the maximum the queue can hold for a class policy configured in a policy map. The default value of the *units* argument is **packets**. In this example, when the queue limit reaches 1,000,000 bytes, enqueued packets to the class queue are dropped.
Step 5	**priority**[**level** *priority-level*] **Example:** `RP/0/0/CPU0:router(config-pmap-c)# priority level 1`	Specifies priority to a class of traffic belonging to a policy map.
Step 6	**police rate percent** *percentage* **Example:** `RP/0/0/CPU0:router(config-pmap-c)# police rate percent 30`	Configures traffic policing.
Step 7	**class** *class-name* **Example:** `RP/0/0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change. In this example, class2 is configured.
Step 8	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class2.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. This example allocates 20 percent of the leftover interface bandwidth to class class2.
Step 10	**exit** **Example:** `RP/0/0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

	Command or Action	Purpose
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface POS 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	**commit**	

Configuring Multicast Egress QoS

This task configures multicast egress QoS and disables multicast egress traffic on the priority queue.

Prerequisites

Multicast egress QoS can be configured only on an interface that is already configured for multicast routing. See *Cisco IOS XR Multicast Configuration Guide for* Cisco XR 12000 Series Router for information on configuring multicast routing.

SUMMARY STEPS

1. **configure**
2. **hw-module qos multicast location** *node-id*
3. **hw-module qos multicast priorityq disable location** *node-id*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**hw-module qos multicast location** *node-id* **Example:** `RP/0/0/CPU0:router(config)# hw-module qos multicast location 0/2/0`	Enables multicast egress QoS for a specified location. This command is needed only for multicast QoS on 2.5 Gbps IP Services Engine (Engine 3) linecards.

	Command or Action	Purpose
Step 3	**hw-module qos multicast priorityq disable location** *node-id* **Example:** `RP/0/0/CPU0:router(config)# hw-module qos multicast priorityq disable location 0/2/0`	Diverts multicast traffic slated for the priority queue to the default queue QoS for a specified location.
Step 4	**commit**	

Configuration Examples for Configuring Policy Maps

This section provides these configuration examples:

Enabling WRED for IPv4 Egress Multicast Traffic

The following example shows a policy map configuration that meets the requirements to enable WRED for IPv4 egress multicast traffic:

```
policy-map policy_A
 class prec0123
   bandwidth percent 20
   random-detect precedence routine 1000 packets 2000 packets
   random-detect precedence priority 1000 packets 4000 packets
   random-detect precedence immediate 2000 packets 5000 packets
 class prec45
   bandwidth percent 40
   random-detect precedence flash-override 4000 packets 8000 packets

policy-map policy_B
 class prec0123
   bandwidth percent 10
   random-detect precedence routine 1000 packets 4000 packets
   random-detect precedence priority 2000 packets 5000 packets
   random-detect precedence immediate 3000 packets 6000 packets
 class prec45
   bandwidth percent 30
   random-detect precedence flash-override 4000 packets 8000 packets

policy-map policy_C
 class prec012
   bandwidth percent 20
 class prec34
   bandwidth percent 40

interface pos0/1/0/0
 service-policy output policy_A
interface pos 0/1/0/1
 service-policy output policy_B
interface pos 0/1/0/3
 service-policy output policy_C
```

The interfaces mapped to policy_A and policy_B are configured with the same classes and RED statements. The precedences for policy_A and policy_B are mapped to the same RED profile although the minimum and

maximum RED thresholds are different. The interface mapped to policy_C is configured differently than the interfaces mapped to policy_A and policy_B.

Additional References

These sections provide references related to implementing QoS congestion avoidance.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco XR 12000 Series Router
Master command reference	Cisco XR 12000 Series Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco XR 12000 Series Router
User groups and task IDs	"*Configuring AAA Services on Cisco IOS XR Software*" module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html