Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers

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 the following 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.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Random Early Detection	yes	yes
Weighted Random Early Detection	yes	yes
Tail Drop	yes	yes

Feature History for Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	The Congestion Avoidance feature was introduced on ASR 9000 Ethernet Line Cards. The Random Early Detection, Weighted Random Early Detection, and Tail Drop features were introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Random Early Detection, Weighted Random Early Detection, and Tail Drop features were supported on the SIP 700 for the ASR 9000.

Contents

Prerequisites for Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers

The following 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 on Cisco ASR 9000 Series Routers

To configure QoS congestion avoidance techniques in this document you must understand the following concepts:

Random Early Detection and TCP

The 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.

Weighted Random Early Detection for Preferential Traffic Handling

WRED provides the ability to define multiple RED profiles within a single class, based on certain match criteria (DSCP, discard class and so on), so that different drop precedences can be configured based on the relative importance of packets. WRED can selectively discard lower priority traffic when the interface begins to get congested and provide differentiated performance characteristics for different classes of service. You can configure WRED to ignore IP precedence when making drop decisions so that nonweighted RED behavior is achieved.

WRED makes early detection of congestion possible and provides for multiple classes of traffic. It also protects against global synchronization. For these reasons, WRED is useful on any output interface in which you expect congestion to occur.

However, WRED is usually used in the core routers of a network, rather than at the edge of the network. 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.

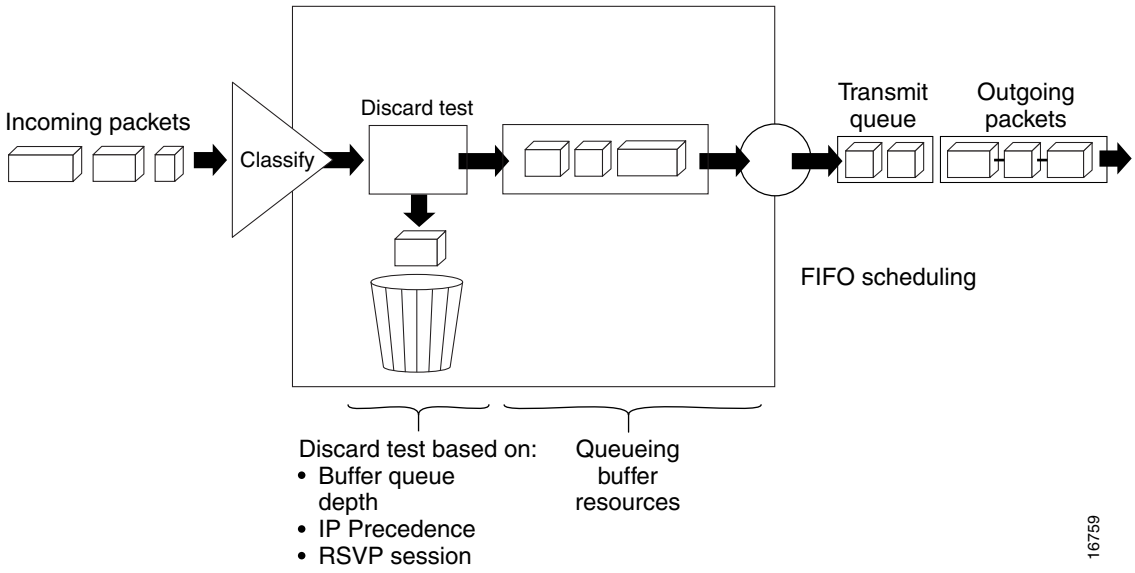
WRED provides separate drop thresholds (minimum and maximum) for different classification criteria (such as IP precedences, MPLS EXP values), allowing you to provide different qualities of service in regard to packet dropping for different traffic types. Standard traffic may be dropped more frequently than premium traffic during periods of congestion.

WRED treats non-IP traffic as precedence 0, the lowest precedence. Therefore, non-IP traffic, in general, is more likely to be dropped than IP traffic.

WRED is useful only when the bulk of the traffic is TCP/IP traffic. With TCP, dropped packets indicate congestion, so the packet source reduces its transmission rate. With other protocols, packet sources may not respond or may resend dropped packets at the same rate. Thus, dropping packets does not decrease congestion.

Figure 1 illustrates how WRED works.

Figure 1 Weighted Random Early Detection



Queue-limit for WRED

Queue-limit is used to fine-tune the number of buffers available for each queue. It can only be used on a queuing class. Default queue limit is 100 ms of the service rate for the given queue. The service rate is the sum of minimum guaranteed bandwidth and bandwidth remaining assigned to a given class either implicitly or explicitly.

The queue-limit is rounded up to one of the following values: 8 KB, 16 KB, 24 KB, 32 KB, 48 KB, 64 KB, 96 KB, 128 KB, 192 KB, 256 KB, 384 KB, 512 KB, 768 KB, 1024 KB, 1536 KB, 2048 KB, 3072 KB, 4196 KB, 8192 KB, 16394 KB, 32768 KB, 65536 KB, 131072 KB, or 262144 KB.

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 and Marking on Cisco ASR 9000 Series Routers" module.

How to Configure Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers

This section contains instructions for the following tasks:

- Configuring Random Early Detection, page 76 (required)
- Configuring Weighted Random Early Detection, page 79 (required)
- Configuring Tail Drop, page 82 (required)

Configuring Random Early Detection

You can configure Random Early Detection (RED) by configuring the **random-detect** command with the **default** keyword on any class. In this way, a single RED profile is applied to all packets matching the class.

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

If you configure the **random-detect default** command on any class including class-default, you must configure one of the following commands:

- **shape average**
- **bandwidth**
- **bandwidth remaining**

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-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. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
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/RSP0/CPU0:router(config-pmap-c)# random-detect default`	Enables RED with default minimum and maximum thresholds.

	Command or Action	Purpose

Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*}
	or
	bandwidth remaining [**percent** *value* | **ratio** *ratio-value*
	
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RSP0/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.
		Note One of these configurations is required for a non-default class.

Step 6	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/0`	Enters configuration mode and configures an interface.

	Command or Action	Purpose
Step 10	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/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	**end**
	or
	commit
	
	Example:
	`RP/0/RSP0/CPU0:router(config-cmap)# end`
	or
	`RP/0/RSP0/CPU0:router(config-cmap)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Weighted Random Early Detection

WRED drops packets selectively based on any specified criteria, such as CoS, DSCP, EXP, discard-class, or precedence . WRED uses these matching criteria to determine how to treat different types of traffic.

Configure WRED using the **random-detect** command and different CoS, DSCP, EXP, and discard-class values. The value can be range or a list of values that are valid for that field. You can also use minimum and maximum queue thresholds to determine the dropping point.

When a packet arrives, the following actions occur:

- If the queue size is less than the minimum queue threshold, the arriving packet is queued.
- If the 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 queue size is greater than the maximum threshold, the packet is dropped.

Restrictions

When configuring the **random-detect dscp** command, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

Only two minimum and maximum thresholds (each with different match criteria) can be configured per class.

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. **shape average** {**percent** | *value* [*units*]}
7. **queue-limit** *value* [*units*]
8. **exit**
9. **interface** *type interface-path-id*
10. **service-policy** {**input** | **output**} *policy-map*
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 4	**random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect dscp af11 1000000 bytes 2000000 bytes`	Changes the minimum and maximum packet thresholds for the DSCP value. • Enables WRED. • *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. 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. 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/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RSP0/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.
		Note One of these configurations is required for a non-default class.

Step 6	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RSP0/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	**queue-limit** *value* [*units*] **Example:** `RP/0/RSP0/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 given queue class.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

	Command or Action	Purpose
Step 9	**interface** *type inteface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	Enters configuration mode and configures an interface.
Step 10	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/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	**end**
	or
	commit
	
	Example:
	`RP/0/RSP0/CPU0:router(config-cmap)# end`
	or
	`RP/0/RSP0/CPU0:router(config-cmap)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

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.

If the GSR for a queue is zero, use the following to compute the default **queue-limit**:

- 1 percent of the interface bandwidth for queues in a nonhierarchical policy.
- 1 percent of minimum parent shape and interface rate for queues within a hierarchical policy.

Note The default **queue-limit** is set to bytes of 100 ms of queue bandwidth. The following formula is used to calculate the default queue limit (in bytes):

bytes = (100 ms / 1000 ms) * queue_bandwidth kbps)) / 8

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. **class** *class-name*
6. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
   or
   bandwidth remaining [**percent** *value* | **ratio** *ratio-value*]
7. **exit**
8. **exit**
9. **interface** *type interface-path-id*
10. **service-policy** {**input** | **output**} *policy-map*
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**queue-limit** *value* [*units*] **Example:** `RP/0/RSP0/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	**class** *class-name* **Example:** `RP/0/RSP0/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 6	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/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 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0`	Enters configuration mode, and configures an interface.
Step 10	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/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.

	Command or Action	Purpose

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RSP0/CPU0:router(config-cmap)# end`
	or
	`RP/0/RSP0/CPU0:router(config-cmap)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Additional References

The following sections provide references related to implementing QoS congestion avoidance.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Master command reference	*Cisco ASR 9000 Series Aggregation Services Router Master Command Listing*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Router" module of *Cisco Cisco ASR 9000 Series Aggregation Services Router 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/techsupport