Congestion Avoidance

Congestion Avoidance

Queuing provides a way to temporarily store data when the received rate of data is larger than what can be sent. Managing the queues and buffers is the primary goal of congestion avoidance. As a queue starts to fill up with data, it is important to try to make sure that the available memory in the ASIC/NPU does not fill up completely. If this happens, subsequent packets coming into the port are dropped, irrespective of the priority that they received. This could have a detrimental effect on the performance of critical applications. For this reason, congestion avoidance techniques are used to reduce the risk of a queue from filling up the memory completely and starving non-congested queues for memory. Queue thresholds are used to trigger a drop when certain levels of occupancy are exceeded.

Scheduling is the QoS mechanism that is used to empty the queues of data and send the data onward to its destination.

Shaping is the act of buffering traffic within a port or queue until it is able to be scheduled. Shaping smoothens traffic, making traffic flows much more predictable. It helps ensure that each transmit queue is limited to a maximum rate of traffic.

Queuing Modes

Two network queuing modes are supported for network interface queuing: the default mode of 8xVOQ (virtual output queuing) and 4xVOQ. To change the mode from one to another requires that you must first reload all line cards in the system.

In the 8xVOQ mode, eight VoQs and their associated resources are allocated for each interface. These queues are allocated regardless of the exact policy configuration on that interface. This mode supports a separate VOQ for each of the eight internal traffic classes.

In the 4xVOQ mode, four VoQs and their associated resources are allocated to each interface, and these queues are allocated regardless of the exact policy applied. In this mode the system supports twice the number of logical interfaces, but the eight traffic classes must be mapped down by configuration to four VoQs, not eight VoQs.

Note From Cisco IOS XR Release 7.2.12 onwards, all the queuing features that are supported on Layer 3 interfaces are also supported on Layer 2 interfaces. However, these features apply only to the main interface (physical and bundle interfaces), and not on the sub-interfaces.

Main Interface Queuing Policy

The main interface default queues are created as part of the main interface creation.

When you apply a queuing policy to the main interface, it will override the default queuing and scheduling parameters for the traffic classes you have configured.

In the 8xVOQ mode, a P1+P2+6PN hierarchy is used for the main interface queues (default queuing and scheduling). The default queues are used for all traffic to the main interface and traffic to any sub-interface without a queuing policy applied. The control/protocol traffic uses traffic class 7 (TC7), priority 1 (P1) to avoid drops during congestion.

Sub-Interface Queuing Policy

Each sub-interface supports up to three policies: an ingress policy, an egress marking policy, and an egress queuing policy. To create and configure a separate set of VoQs for a sub-interface, apply a queuing policy on that sub-interface. When you remove the sub-interface queuing policy, the associated VoQs are freed and the sub-interface traffic reverts to using the main interface VoQs.

Congestion Avoidance in VOQ

Congestion avoidance within a VOQ block is done by applying a congestion management profile to a VOQ. This profile defines the admission criteria and checks performed at the enqueue time. Under normal traffic conditions the packet is enqueued into the Shared Memory System (SMS) buffers. (The shared memory system is the primary packet storage area.) If the SMS VOQ is congested beyond a set threshold, the VOQ is moved to the external High Band Memory (HBM) block. When the HBM queue drains, it is returned to the on-chip SMS. The queue size in HBM is adaptive and decreases when the total HBM usage is high.

Note Random Early Detect (RED) is available only for VOQs in HBM. The hardware does not support Weighted Random Early Detect (WRED).

Sharing of VOQ Statistics Counters

Every network processor on the router has multiple slices (or pipelines), and every slice has a set of VOQs associated with every interface on the router. To maintain counters at high packet rates, two sets of counters are associated with each interface on each network slice. As an example, consider a device with six slices (12 interfaces), each with 24,000 VOQs, where you want both transmitted and dropped events counted. In this scenario, you would require 12 x 24, 000 x 2 = 5, 76,000 counters, which alone exceeds the counter capacity of the device.

It is to mitigate such a scenario that the router supports configurable sharing of VOQ counters. You can configure the sharing such that a counter is shared by {1,2,4,8} VOQs.

Each set of VoQs sharing counters has two counters that measure:

- Enqueued packets count in packets and bytes units.
- Dropped packets count in packets and bytes units.

For the feature to take effect:

- Delete egress queuing policy-map configuration from all interfaces.
- Run the command **# reload location all** to reload all the nodes on your router.

Configuring Sharing of VOQ Statistics Counters

To configure VOQs sharing counters, use the #hw-module profile stats voqs-sharing-counters and specify the number of VOQ counters for each queue.

```
RP/0/RP0/CPU0:ios(config)#hw-module profile stats ?
  voqs-sharing-counters  Configure number of voqs (1, 2, 4) sharing counters
RP/0/RP0/CPU0:ios(config)#hw-module profile stats voqs-sharing-counters ?
  1  Counter for each queue
  2  2 Queues share counters
  4  4 Queues share counters
RP/0/RP0/CPU0:ios(config)#hw-module profile stats voqs-sharing-counters 1
RP/0/RP0/CPU0:ios(config)#hw-module profile stats voqs-sharing-counters 2
RP/0/RP0/CPU0:ios(config)#commit
RP/0/RP0/CPU0:ios#reload location all
```

Running Configuration

```
RP/0/RP0/CPU0:ios#show run | in hw-mod
Mon Feb 10 13:57:35.296 UTC
Building configuration...
hw-module profile stats voqs-sharing-counters 2
RP/0/RP0/CPU0:ios#
```

Verification

```
RP/0/RP0/CPU0:ios#show controllers npu stats voq ingress interface hundredGigE 0/0/0/16
instance all location 0/RP0/CPU0
Mon Feb 10 13:58:26.661 UTC

Interface Name    =   Hu0/0/0/16
Interface Handle  =      f0001b0
Location          =   0/RP0/CPU0
Asic Instance     =            0
VOQ Base          =        10288
```

```
Port Speed(kbps)  =    100000000
Local Port        =        local
VOQ Mode          =            8
Shared Counter Mode =          2
       ReceivedPkts    ReceivedBytes   DroppedPkts     DroppedBytes
-------------------------------------------------------------------
TC_{0,1} = 114023724       39908275541     113945980       39881093000
TC_{2,3} = 194969733       68239406550     196612981       68814543350
TC_{4,5} = 139949276       69388697075     139811376       67907466750
TC_{6,7} = 194988538       68242491778     196612926       68814524100
```

Related Commands

hw-module profile stats voqs-sharing-counters

Dual Queue Limit

The dual queue limit option is added to **queue-limit** command on the CLI of your router and displays as **discard-class**. What the **discard-class** option does is give you the flexibility to configure two queue limits on a single policy map—one for high-priority traffic and the other for low-priority traffic. This option ensures that the high priority traffic flow continues unaffected (up to the derived threshold from the **discard-class 0** queue-limit) while the low-priority traffic continues up to the lower threshold (per **discard-class 1** queue-limit).

Tell Me More

You can configure the two queue limits per these details:

- One for flow that you mark as **discard-class 0** (higher priority) on ingress via ingress-policy.
- second, for flow that you mark as **discard-class 1** (lower priority) on ingress via ingress policy.

The **discard-class 1** flow (for low-priority traffic) begins to drop when the queue length hits the size limit that you configured for discard-class 1. Conversely, the flow for **discard-class 1** stops dropping when queue-length falls below its configured value.

As an example, consider this configuration:

```
policy-map egress_pol_dql
class tc7
  queue-limit discard-class 0 100 mbytes
  queue-limit discard-class 1 50 mbytes
  priority level 1
 !
 class class-default
  bandwidth remaining ratio 1
!
end-policy-map
!
```

Also consider the verification:

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios#show qos interface hundredGigE 0/0/0/30 output
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/30 ifh 0xf000210  -- output policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   egress_pol_dql
VOQ Base:                      464
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
VOQ Mode:                      8
```

```
Shared Counter Mode:           1
------------------------------------------------------------------------------
Level1 Class (HP1)                       =   tc7
Egressq Queue ID                         =   471 (HP1 queue)
Queue Max. BW.                           =   no max (default)
Discard Class 1 Threshold                =   25165824 bytes / 2 ms (50 mbytes)
Discard Class 0 Threshold                =   75497472 bytes / 5 ms (100 mbytes)
WRED not configured for this class

Level1 Class                             =   class-default
Egressq Queue ID                         =   464 (Default LP queue)
Queue Max. BW.                           =   no max (default)
Inverse Weight / Weight                  =   1 / (1)
TailDrop Threshold                       =   749568 bytes / 6 ms (default)
WRED not configured for this class
```

In the preceding example, there are two traffic flows that are marked as **discard-class 0** (higher priority) and **discard-class 1** (lower priority).

As long as the queue length of the two flows remains below 25165824 bytes (the threshold for **discard-class 1**), packets from both flows continue without any drops. When the queue length reaches 25165824 bytes, **discard-class 1** packets are not enqueued, ensuring all remaining bandwidth is used for the higher priority flow (**discard-class 0**).

The higher priority flow drops only when the queue length reaches 75497472 bytes.

Note

- This option protects the high-priority traffic from loss due to congestion, but not necessarily from latency due to congestion.
- These thresholds are derived from hardware-specific queue regions.

Restrictions

Ensure that you read these restrictions about the dual queue limit option.

- Both the queue-limits must use the same unit of measurement.
- The queue limit for **discard-class** *0* must always be greater than that for **discard-class** *1*.
- When the discard-class option is not used to configure the queue-limit, packets marked with **discard-class** *0* and **discard-class** *1* have the same queue-limit; in other words, they receive identical treatment.
- A queue-limit that is configured with only **discard-class** *0* or **discard-class** *1* is rejected.

Equitable Traffic Flow Using Fair VOQ

Table 1: Feature History Table

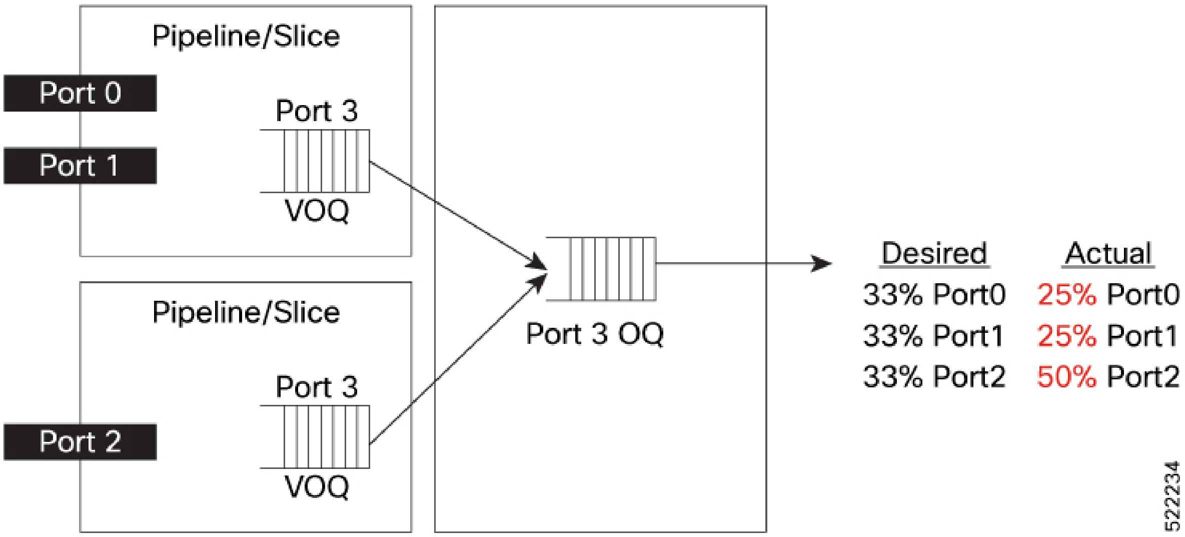
Feature Name	Release Information	Feature Description
Equitable Traffic Flow Using Fair VOQ	Release 7.3.3	Configuring this feature ensures that ingress traffic from various source ports on every network slice of an NPU is assigned a unique virtual output queue (VOQ) for every source port and destination port pair. This action ensures that the bandwidth available at the destination port for a given traffic class is distributed equally to all source ports requesting bandwidth. In earlier releases, the traffic wasn't distributed equitably because each slice wasn't given its fair share of the output queue bandwidth. This feature introduces the **fair-4** and **fair-8** keywords in the **hw-module profile qos voq-mode** command.
Equitable Traffic Flow Using Fair VOQ on Cisco 8201-32FH Routers	Release 7.5.2	You can now ensure that the bandwidth available at the destination port for a given traffic class is distributed equally to all source ports requesting bandwidth on Cisco 8201-32FH routers.

Fair VOQ: Why

Per default behavior, every network slice of an NPU is assigned a set of 4 or 8 Virtual Output Queues (VOQ) per destination port. With such an assignment, it's challenging to ensure that the right amount of buffering is available via VOQs. With this configuration, the ingress traffic from various source ports on a slice (or pipeline) on an NPU destined to a destination port is assigned to a VOQ per slice. In other words, multiple source ports sending traffic to the same destination port use the same VOQ. However, when sending traffic to different destination ports, traffic is enqueued to different VOQs. This means that the traffic isn't distributed equitably because each slice doesn't get its fair share of the output queue bandwidth. In a scenario where one slice has two ports and another slice has only one port, the bandwidth drops for the ports sharing a slice, even though the two ports handle more traffic than the single port.

Consider the following example where two 100G ports—port-0 and port-1—that belong to the same slice (slice-0) are sending traffic to port-3 on the output queue (OQ). You have a 100G port on another slice (slice-1) on the same NPU that is also scheduled to send traffic to port-3. The ingress VOQ is shared between the two ports in slice-0, whereas the ingress VOQ in slice-1 is available exclusively for port-3. This arrangement results in port-0 and port-1 getting 25% of the buffer traffic, while port-3 receives 50% of the buffer traffic.

Figure 1: Existing behavior : Source ports on slice share one VOQ per destination port



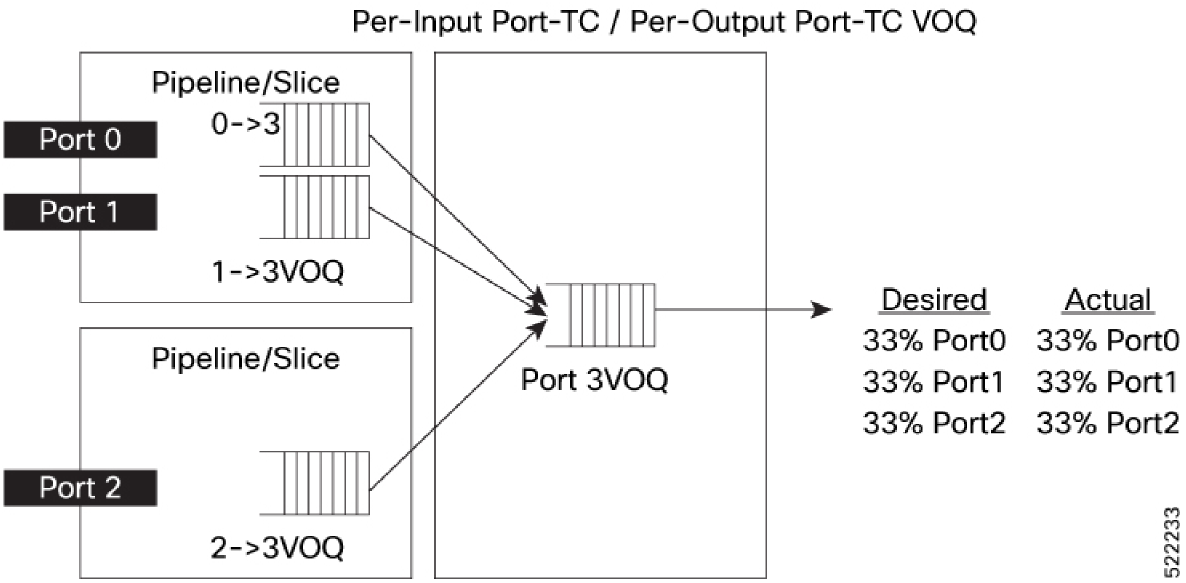
The fair VOQ feature solves this disparity in traffic distribution.

Fair VOQ: How

The fair VOQ feature tackles the default behavior that treats source ports on each NPU slice equally, regardless of the number of active source ports. It does so by redesigning the way bandwidth is allocated from the output queue. Rather than distributing bandwidth at a slice level, fair VOQ distributes bandwidth directly to the source ports. When you configure the command **hw-module profile qos voq-mode** and reload your router, the functionality creates a dedicated VOQ for every source port and destination port pair. This arrangement ensures that bandwidth available at the destination port for a given traffic class is distributed equally to all source ports requesting bandwidth.

Extending the preceding example to understand the fair VOQ functionality, there are now dedicated VOQs for each ingress port that connect to the port on the output queue. Thus, port-0 and port-1 now don’t share a VOQ, and port-3 has its VOQ as before, as shown in the following figure. This fair VOQ arrangement results in traffic queued on dedicated queues, thus improving the traffic performance.

Figure 2: Fair VOQ behavior: each source port on slice has one dedicated VOQ per destination port



Fair VOQ Modes and Sharing of Counters

You can configure fair VOQ for 8xVOQ mode (fair-8) and 4xVOQ mode (fair-4) using the following options in the **hw-module profile qos voq-mode** command:

- **hw-module profile qos voq-mode fair-8**
- **hw-module profile qos voq-mode fair-4**

You can also share VOQ statistics counters in both fair VOQ modes, as shown in the following table. (For details on why sharing counters is essential and how to configure counters sharing, see Sharing of VOQ Statistics Counters, on page 3.)

Table 2: Fair VOQ Modes and Sharing Counters

Fair VOQ Mode	Sharing Counters Mode	Important Notes
fair-8	2, 4	• Eight VOQs configured per source port and destination pair • Counters are shared by {2, 4} VOQs. • **fair-8** mode doesn't support dedicated counter mode (counter mode1, where there's a counter for every queue)

Fair VOQ Mode	Sharing Counters Mode	Important Notes
fair-4	1, 2, 4	• Four VOQs configured per source port and destination pair • Counters are shared by {1, 2, 4} VOQs.

Fair VOQs and Slice (or Normal) VOQs: Key Differences

The following table is a snapshot to outline the key differences between fair VOQs and slice or regular VOQs.

Table 3: Fair VOQs and Normal VOQs

Fair VOQ	Normal VOQ
fair-8 mode: eight VOQs configured per source port and destination pair	**8**: • Eight VOQs per destination port per slice • These VOQs are shared by all source ports within an NPU slice.
fair-4 mode: four VOQs configured per source port and destination pair	**4**: • Four VOQs per destination port per slice • These VOQs are shared by all source ports within an NPU slice.

Guidelines and Limitations

- The fair VOQ feature is supported on the Cisco 8202 router (12 QSFP56-DD 400G and 60 QSFP28 100G ports) and Cisco 8201-32FH router (32 QSFP-DD 400G ports).
- The following table details the maximum interfaces (with basic IPv4 configurations and no other scale configuration such as QoS policy, ACL, and subinterface configuration) allowed based on VOQ mode and sharing counter mode.

Table 4: Maximum Interfaces Based on Fair VOQ Mode and Sharing Counter Mode

VOQ Mode	Sharing Counter Mode	Maximum Interfaces for Cisco 8202	Maximum Interfaces for Cisco 8201-32FH
fair-8	1	**The router doesn't support this combination.**	**The router doesn't support this combination.**

VOQ Mode	Sharing Counter Mode	Maximum Interfaces for Cisco 8202	Maximum Interfaces for Cisco 8201-32FH
fair-8	2	**96** = 60 (100G) + 8x4 + 4 (400G) ==> you can configure only eight 400G interfaces in 4x10G or 4x25G breakout mode.	**128** = 32x4 (4x10G or 4x25G - breakout on all 32 ports - 400G)
fair-8	4	**108 = 60 + 12 x 4 (breakout on all 12 ports - 400G)**	
fair-4	1	**96** = 60(100G) + 8x4 + 4 (400G) ==> you can configure only eight 400 G interfaces in 4x10G or 4x25G breakout mode.	
fair-4	2	**108** = 60 + 12 x4 (breakout on all 12 ports - 400G)	
fair-4	4	**108** = 60 + 12 x4 (breakout on all 12 ports - 400G)	

Note We recommend using sharing counter mode 4 in breakout modes and sharing counter mode 2 for nonbreakout modes.

If the counter usage of applications is high, sharing counter mode 2 is not supported.

Note Breakout mode isn't supported on 100G interfaces.

- Ensure that you reload the router for the configuration to take effect.
- Layer 2 traffic isn't supported in **fair-voq** mode (**fair-4** and **fair-8**).
- Subinterface queueing isn't supported. (This applies to bundle sub-interfaces as well). This means that you can't attach egress service-policies that require dedicated VOQs. However, egress marking is supported for subinterfaces.
- ERSPAN mirroring traffic is not supported in **fair-voq** mode.
- **hw-module profile stats voqs-sharing-counters** 1 isn't supported in **fair-8** mode. Ensure that you configure **hw-module profile voq sharing-counters** 2 or **hw-module profile voq sharing-counters** 4

along with **hw-module profile qos voq-mode fair-4** or **hw-module profile qos voq-mode fair-8** before reloading the router.

- Breakout is supported only on 400G interfaces in **fair-voq** mode (both **fair-4** and **fair-8**) on the Cisco 8202 router.
- **src-interface** and **src-slice** keywords in **show controller npu stats** are visible only when you configure the VOQ mode to either **fair-8** or **fair-4**.

Configure Fair VOQ

To configure fair VOQ:

1. Configure sharing of VOQ statistics counters. This example configures 2 counters.

Note Configuring **fair-8** mode without counter sharing may cause configuration failure or other unexpected behavior.

2. Configure fair VOQ mode. This example shows how to configure **fair-8** mode.
3. Restart the router for the configuration to take effect.
4. You have successfully enabled the fair VOQ feature to ensure equitable traffic distribution between every source port and destination port pair.

```
/*Configure sharing of VOQ statistics counters; we're configuring 2 counters per queue*/
Router(config)#hw-module profile stats ?
  voqs-sharing-counters  Configure number of voqs (1, 2, 4) sharing counters
Router(config)#hw-module profile stats voqs-sharing-counters ?
  1  Counter for each queue
  2  2 Queues share counters
  4  4 Queues share counters
Router(config)#hw-module profile stats voqs-sharing-counters 2

/*Configure fair-voq mode; we're configuring fair-8 VOQ mode here*/
Router#config
Router(config)#hw-module profile qos voq-mode fair-8
Router(config)#commit
Router#reload location all
```

Running Configuration

```
hw-module profile stats voqs-sharing-counters 2
!
hw-module profile qos voq-mode fair-8
!
```

Verification

Run the **show controller npu stats voq ingress interface** <> **instance** <> **location** <> command to verify the fair VOQ configuration.

```
Router#show controllers npu stats voq ingress interface hundredGigE 0/0/0/20 instance 0
location 0/RP0/CPU0


Interface Name        =   Hu0/0/0/20
Interface Handle      =      f000118
Location              =    0/RP0/CPU0
Asic Instance         =            0
Port Speed(kbps)      =    100000000
Local Port            =        local
Src Interface Name    =          ALL
VOQ Mode              =       Fair-8
Shared Counter Mode   =            2
       ReceivedPkts    ReceivedBytes   DroppedPkts     DroppedBytes
-------------------------------------------------------------------
TC_{0,1} = 11110           1422080         0               0
TC_{2,3} = 0               0               0               0
TC_{4,5} = 0               0               0               0
TC_{6,7} = 0               0               0               0
RP/0/RP0/CPU0:ios#
```

Associated Commands

hw-module profile qos voq-mode

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. The router supports these QoS congestion avoidance techniques:

- Tail Drop and the FIFO Queue, on page 12
- Random Early Detection and TCP, on page 14

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 placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

Configure 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, the enqueued packets to the class queue result in tail drop (packet drop).

Restrictions

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

Configuration Example

You have to accomplish the following to complete the tail drop configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Specifying the maximum limit the queue can hold for a class policy configured in a policy map.
4. Specifying priority to a class of traffic belonging to a policy map.
5. (Optional) Specifying the bandwidth allocated for a class belonging to a policy map or specifying how to allocate leftover bandwidth to various classes.
6. Attaching a policy map to an output interface to be used as the service policy for that interface.

```
Router# configure
Router(config)# policy-map test-qlimit-1
Router(config-pmap)# class qos-1
Router(config-pmap-c)# queue-limit 100 us
Router(config-pmap-c)# priority level 7
Router(config-pmap-c)# exit
Router(config-pmap)# exit

Router(config)# interface HundredGigE 0/6/0/18
Router(config-if)# service-policy output test-qlimit-1
Router(config-if)# commit
```

Running Configuration

```
policy-map test-qlimit-1
 class qos-1
  queue-limit 100 us
  priority level 7
 !
 class class-default
 !
 end-policy-map
!
```

Verification

```
Router# show qos int hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/6/0/18 ifh 0x3000220  -- output policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11176
VOQ Stats Handle:              0x88550ea0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class (HP7)                      =   qos-1
Egressq Queue ID                        =   11177 (HP7 queue)
```

```
TailDrop Threshold                       =   1253376 bytes / 100 us (100 us)
WRED not configured for this class

Level1 Class                             =   class-default
Egressq Queue ID                         =   11176 (Default LP queue)
Queue Max. BW.                           =   101803495 kbps (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
TailDrop Threshold                       =   1253376 bytes / 10 ms (default)
WRED not configured for this class
```

Related Topics

- Tail Drop and the FIFO Queue, on page 12

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. It achieves this by taking action on the average queue size, and not the instantaneous queue size. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

Configure Random Early Detection

The **random-detect** command with the minimum threshold and maximum threshold keywords must be used to enable random early detection (RED).

Guidelines

- If you configure the **random-detect <min threshold> <max threshold>** command on any class including class-default, configure one of the following commands: **shape average** or **bandwidth remaining**.
- If you configure a queue-limit that is lesser than the minimum supported value, the configured value automatically adjusts to the supported minimum value.

  While configuring **random-detect**, if you set the **<min threshold>** and **<max-threshold>** values lesser than the minimum supported threshold value:

  - The **<min threshold>** value automatically adjusts to the minimum supported value.
  - The **<max-threshold>** value doesn’t autoadjust to a value above the minimum supported threshold value. This results in a failed **random-detect** configuration. To prevent this error, configure the **<max-threshold>** value such that it exceeds the **<min threshold>** value that your system supports.

Configuration Example

Accomplish the following to complete the random early detection configuration:

1. Creating (or modifying) a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Enabling RED with minimum and maximum thresholds.
4. Configure **one** of the following:
   - Specifying how to allocate leftover bandwidth to various classes. **OR**
   - Shaping traffic to the specified bit rate or a percentage of the available bandwidth.
5. Attaching a policy map to an output interface to be used as the service policy for that interface.

```
Router# configure
Router(config)# policy-map red-abs-policy
Router(config-pmap)# class qos-1
Router(config-pmap-c)# random-detect <min threshold> <max threshold>
Router(config-pmap-c)# shape average percent 10
Router(config-pmap-c)# end-policy-map
Router(config)# commit
Router(config)# interface HundredGigE0/0/0/12
Router(config-if)# service-policy output red-abs-policy
Router(config-if)# commit
```

Running Configuration

```
policy-map red-abs-policy
class tc7
  priority level 1
  queue-limit 75 mbytes
 !
 class tc6
  priority level 2
  queue-limit 75 mbytes
 !
 class tc5
  shape average 10 gbps
  queue-limit 75 mbytes
 !
 class tc4
  shape average 10 gbps
  queue-limit 75 mbytes
 !
 class tc3
  shape average 10 gbps
  queue-limit 75 mbytes
 !
 class tc2
  shape average 10 gbps
  queue-limit 75 mbytes
 !
 class tc1
  shape average 10 gbps
  random-detect ecn
  random-detect 100 mbytes 200 mbytes
 !
 class class-default
  shape average 10 gbps
  random-detect 100 mbytes 200 mbytes
 !
 end-policy-map
!

interface HundredGigE0/0/0/12
service-policy output red-abs-policy
shutdown
!
```

Verification

```
Router# show qos int hundredGigE 0/6/0/18 output

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/12 ifh 0x3000220  -- output policy
NPU Id:                        3
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
VOQ Base:                      11176
VOQ Stats Handle:              0x88550ea0
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   qos-1
Egressq Queue ID                         =   11177 (LP queue)
Queue Max. BW.                           =   10082461 kbps (10 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
Guaranteed service rate                  =   10000000 kbps
TailDrop Threshold                       =   12517376 bytes / 10 ms (default)

Default RED profile
RED Min. Threshold                      =   12517376 bytes (10 ms)
RED Max. Threshold                      =   12517376 bytes (10 ms)

Level1 Class                             =   class-default
Egressq Queue ID                         =   11176 (Default LP queue)
Queue Max. BW.                           =   101803495 kbps (default)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
Guaranteed service rate                  =   50000000 kbps
TailDrop Threshold                       =   62652416 bytes / 10 ms (default)
WRED not configured for this class
```

Related Topics

- Random Early Detection and TCP, on page 14

Explicit Congestion Notification

Random Early Detection (RED) is implemented at the core routers of a network. Edge routers assign IP precedences to packets, as the packets enter the network. With RED, core routers then use these precedences to determine how to treat different types of traffic. RED provides a single threshold and weights per traffic class or queue for different IP precedences.

ECN is an extension to RED. ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured, ECN helps routers and end hosts to understand that the network is congested and slow down sending packets. However, if the queue length is above the maximum threshold for the extended memory, packets are dropped. This is the identical treatment that a packet receives when RED is enabled without ECN configured on the router.

RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*, states that with the addition of active queue management (for example, RED) to the Internet infrastructure, routers are no longer limited to packet loss as an indication of congestion.

Note You cannot use this feature when you have set qos-group or mpls experimental along with a traffic class in the ingress policy.

Implementing ECN

Implementing ECN requires an ECN-specific field that has two bits—the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit—in the IP header. The ECT bit and the CE bit can be used to make four code points of 00 to 11. The first number is the ECT bit and the second number is the CE bit.

Table 5: ECN Bit Setting

ECT Bit	CE Bit	Combination Indicates
0	0	Not-ECN-capable.
0	1	Endpoints of the transport protocol are ECN-capable.
1	0	Endpoints of the transport protocol are ECN-capable.
1	1	Congestion experienced.

The ECN field combination 00 indicates that a packet is not using ECN. The code points 01 and 10—Called ECT(1) and ECT(0), respectively—are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two code points identically. Data senders can use either one or both of these two combinations. The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

Packet Handling When ECN Is Enabled

When ECN is enabled, all packets between <min_threshold> and <max tail drop threshold> are marked with ECN. Three different scenarios arise if the queue length is between the minimum threshold and the maximum threshold:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)—and the RED algorithm determines that the packet should have been dropped based on the drop probability—the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.

- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet is transmitted. If, however, the max tail drop threshold is exceeded, the packet is dropped. This is the identical treatment that a packet receives when RED is enabled without ECN configured on the router.
- If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

Configuration Example

```
Router# configure
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth percent 50
Router(config-pmap-c)# random-detect 1000 packets 2000 packets
Router(config-pmap-c)# random-detect ecn
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# commit
```

Verification

Use the **show policy-map interface** to verify the configuration.

```
Router# show policy-map int hu 0/0/0/35 output
TenGigE0/0/0/6 output: pm-out-queue

HundredGigE0/0/0/35 output: egress_qosgrp_ecn

Class tc7
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           195987503/200691203072         0
    Transmitted         :           188830570/193362503680         0
    Total Dropped       :             7156933/7328699392           0
  Queueing statistics
    Queue ID                             : 18183
    Taildropped(packets/bytes)           : 7156933/7328699392

    WRED profile for
    RED Transmitted (packets/bytes)            : N/A
    RED random drops(packets/bytes)            : N/A
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): 188696802/193225525248
Class tc6
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           666803815/133360763000         0
    Transmitted         :           642172362/128434472400         0
    Total Dropped       :            24631453/4926290600           0
  Queueing statistics
    Queue ID                             : 18182
    Taildropped(packets/bytes)           : 24631453/4926290600

    WRED profile for
    RED Transmitted (packets/bytes)            : N/A
    RED random drops(packets/bytes)            : N/A
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): 641807908/128361581600
```

```
Class tc5
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           413636363/82727272600          6138
    Transmitted         :           398742312/79748462400          5903
    Total Dropped       :            14894051/2978810200           235
  Queueing statistics
    Queue ID                             : 18181
    Taildropped(packets/bytes)           : 14894051/2978810200

    WRED profile for
    RED Transmitted (packets/bytes)            : N/A
    RED random drops(packets/bytes)            : N/A
    RED maxthreshold drops(packets/bytes)      : N/A
    RED ecn marked & transmitted(packets/bytes): 398377929/79675585800
```

Note The **RED ecn marked & transmitted(packets/bytes)** row displays the statistics for ECN marked packets. To begin with, it displays *0/0*.

Monitor Congestion and Delay High Water Marks for Traffic Class Queues