Modular QoS Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.7.x

First Published: 2022-07-01

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

This preface contains these sections:

- Changes to This Document, on page vii
- Communications, Services, and Additional Information, on page vii

Changes to This Document

This table lists the technical changes made to this document since it was first published.

Table 1: Changes to this Document

Date	Change Summary
July 2022	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

CHAPTER 1

New and Changed QoS Features

New and Changed QoS Features

Table 2: QoS Features Added or Modified in IOS XR Release 7.7.x

Feature	Description	Changed in Release	Where Documented
NA	NA	NA	NA

CHAPTER 2

Traffic Management Overview

Scope

Read this configuration guide to understand the overall architecture that powers the Cisco Quality of Service (QoS) technology, and also how to use its features to configure and manage the traffic bandwidth and packet loss parameters on your network.

Traditional Traffic Management

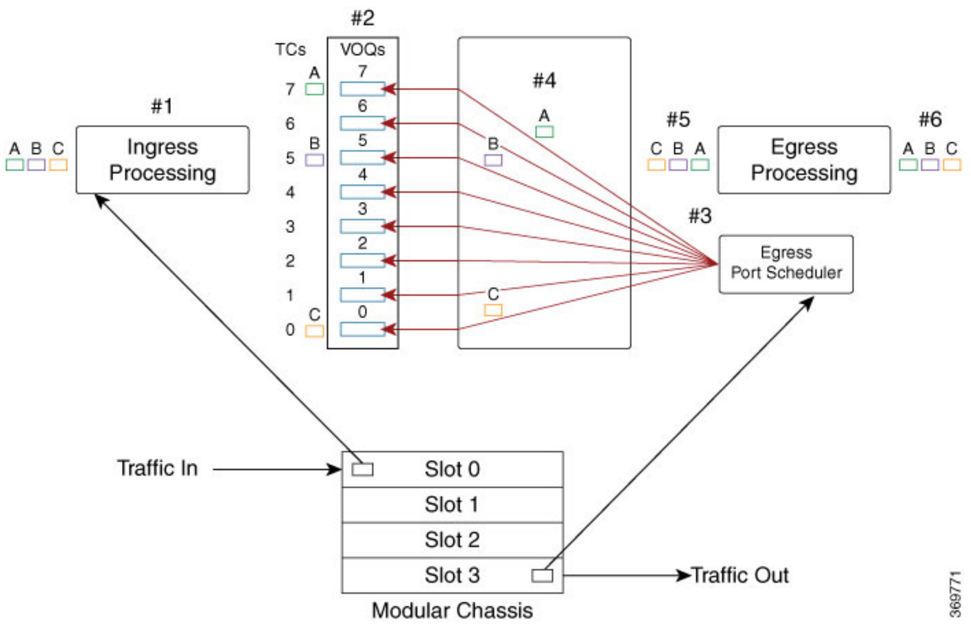
In traditional methods of traffic management, traffic packets are sent to the egress output queues without taking into consideration the egress interface availability to transmit.

Therein lies the problem as well. In case of a traffic congestion, the traffic packets may get dropped at the egress port. Which means that the network resources spent getting the packets from the ingress input queue across the switch fabric to the output queues at egress are wasted. That is not all—input queues buffer traffic meant for different egress ports, so congestion on one egress port could affect traffic on another port, an event referred to as head-of-line-blocking.

Traffic Management on Your Router

Your router's Network Processing Unit (NPU) uses a coupled ingress-egress virtual output queuing (VoQ)-based forwarding architecture to manage traffic.

Figure 1: Traffic flow from ingress port on slice 0 to an egress port on slot 3



Here, each ingress traffic class has a one-to-one VoQ mapping from each ingress slice (pipeline) to each egress port. Which means that every egress interface (#5 in the figure) has earmarked buffer space on every ingress pipeline (#1 in the figure) for each of its VoQs.

Here is how the packet travels in times of congestion on your router system:

#1: Packets A (colored green), B (colored pink), and C (colored brown) are at the ingress interface. This is where packet marking, classification, and policing takes place. (For details, see Mark Packets to Change Priority Settings, on page 25, Classify Packets to Identify Specific Traffic, on page 9, and Congestion Management, on page 69 .)

#2: These packets are stored in separate buffer storage spaces in dedicated VoQs. This is where queuing, VoQ transmit, and drop packet and byte counters come into play. (For details, see Congestion Avoidance, on page 41 .)

#3: Depending on the bandwidth available on the egress interface, these packets are subjected to egress scheduling, where egress credit and transmit schedulers are configured. In other words, the packets and the sequence in which they will now proceed towards the egress interface is determined here. This is where the fabric bandwidth is taken into consideration for egress scheduling.

#4: The packets are switched through the fabric.

#5: In the final phase, the egress marking and classification takes place, and the congestion is managed in a way that at this stage there is no packet dropped, and all the packets are transmitted to the next hop.

Limitations of the VoQ Model

While the VoQ model of traffic management offers distinct advantages (reducing memory bandwidth requirements, providing end-to-end QoS flow), it has this limitation:

The total egress queue scale is lower because each egress queue must be replicated as an ingress VoQ on each slice of each NPU/ASIC in the system. This means that with the addition of 1 NPU with 20 interfaces, the number of VoQs used on each and every NPU in the system will increase by 20 x 8 (queue/interface) = 160. There is also an increase in the number of credit connectors from each scheduler for each egress port on pre-existing NPUs to each slice in the newly inserted NPU.

QoS Policy Inheritance

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
QoS Policy Inheritance	Release 7.3.15	To create QoS policies for subinterfaces, you had to apply the policy on each subinterface manually. From this release, all you do is create and apply a single QoS policy on the main interface and the subinterfaces automatically inherit the policy. The inheritance model provides an easily maintainable method for applying policies, enabling you to create targeted policies for a group of interfaces and their subinterfaces. This model saves your time and resources while creating QoS policies.

- **What is this functionality all about?**—As the name suggests, the functionality is based on an inheritance model, where you create and apply a QoS policy to a main interface. The subinterfaces that are attached to the main interface automatically inherit the policy.

  The inheritance model applies to all QoS operations including:

  - Classification
  - Marking
  - Policing
  - Shaping

- **How does the inheritance model help?**—Previously, if you had, for example, eight subinterfaces, you created and applied policies to each of those subinterfaces separately. With the inheritance model, you save on time and resources, with just one policy automatically applied across a main interface and its subinterfaces.

- **Do I need to do anything more to enable the inheritance model?**—No, you don't. The inheritance model is the default option.

- **What if I want to override the inheritance option?**—Technically, you can't override this option. However, you can remove the policy from main interface and add policies to the subinterfaces except to the ones where you don't want the policy to be inherited.

- **What about policy-map statistics?**—There's no change to this behavior. Running the show policy-map interface command displays the cumulative statistics for an interface, and these numbers include the subinterfaces as well.
- **Any limitations I need to be aware of?**—There's no support for ECN marking and egress marking policy on the same interface and subinterface combination. However, the QoS policy inheritance functionality accepts these multiple policies causing the ECN markings to fail. To prevent such failures:
    - Don't configure an egress marking policy on a subinterface and apply an ECN-enabled policy on the main interface.
    - Don't apply an ECN policy on a subinterface and configure an egress marking policy on the main interface.

Cisco Modular QoS CLI to Deploy QoS

Cisco Modular QoS CLI (MQC) framework is the Cisco IOS QoS user language that enables:

- a standard Command Line Interface (CLI) and semantics for QoS features.
- simple and accurate configurations.
- QoS provisioning within the context of an extensible language.

For your router, in the egress direction, two types of MQC policies are supported: queuing and marking. You use the queuing policy to configure credit scheduling hierarchy, rates, priority, buffering, and congestion avoidance. You use the marking policy to classify and mark packets that have been scheduled for transmission. Even when a queuing policy is not applied, there is an implicit queuing policy with TC7 - P1, TC6 - P2, TC5 - TC0 (6 x Pn), so packets marked with TC7 and control inject packets are always prioritized over other packets. In the ingress, only one policy is supported for classification and marking.

You can apply queuing and marking policy independent of each other or together in the egress direction. If you apply both policies together, the queuing policy actions are provisioned first, followed by marking policy actions.

Important Points about MQC Egress Queuing Policy

These are important points that you must know about the MQC egress queuing policy:

- The MQC queuing policy consists of a set of class maps, which are added to a policy map. You control queuing and scheduling parameters for that traffic class by applying actions to the policy.
- **class-default** matches **traffic-class 0**, and no other class can match **traffic-class 0**. When **set traffic-class** is used in the ingress policy-map, then it should match the **traffic-class** in the egress queueing policy-map to have a predictable behaviour.
- Each unique combination of traffic classes that match **class-default** require a separate traffic class (TC) profile. The number of TC profiles are limited to 8 for main interfaces and 8 for sub-interfaces.
- You cannot configure multiple traffic classes with the same priority level.
- Each priority level, when configured, must be configured to the class that matches the corresponding TC as shown in the following table.

Priority Level	Traffic Class
P1	7
P2	6
P3	5
P4	4
P5	3
P6	2
P7	1

- If all the priority levels configured in a policy-map are sorted, they must be contiguous. In other words, you cannot skip a priority level. For example, P1 P2 P4 (skipping P3), is not allowed.
- From IOS XR Release 7.3.1 onwards, you can create a single set of contiguous priority TCs. Ensure that you assign priority levels that increase for every TC and that you assign priority level 1 for traffic class 7. You need not configure unused traffic classes, so you can create only those many TCs that you require on the egress policy-map.
- MQC supports up to two levels (parent, child) of queuing policy. The parent level aggregates all the traffic classes and whereas the child level differentiates traffic classes using MQC classes.
- Only these actions are supported in the queuing policy:
  - priority
  - shape
  - bandwidth remaining ratio
  - queue-limit
  - Random Early Detection (RED)
  - Priority flow control
- You can have only one match traffic-class value in the class map.
- You cannot apply a queuing policy to a main interface and its sub-interfaces.

CHAPTER 3

Classify Packets to Identify Specific Traffic

Classify Packets to Identify Specific Traffic

Read this section to get an overview of packet classification and the different packet classification types for your router.

Packet Classification Overview

Packet classification involves categorizing a packet within a specific group (or class) and assigning it a traffic descriptor to make it accessible for QoS handling on the network. The traffic descriptor contains information about the forwarding treatment (quality of service) that the packet should receive. Using packet classification, you can partition network traffic into multiple priority levels or classes of service.

When traffic descriptors are used to classify traffic, the source agrees to adhere to the contracted terms and the network promises a quality of service. This is where traffic policers and traffic shapers come into the picture. Traffic policers and traffic shapers use the traffic descriptor of a packet—that is, its classification—to ensure adherence to the contract.

The Modular Quality of Service (QoS) command-line interface (MQC) is used to define the traffic flows that must be classified, where each traffic flow is called a class of service, or class. Later, a traffic policy is created and applied to a class. All traffic not identified by defined classes fall into the category of a default class.

QoS Support for VPWS

Table 4: Feature History Table

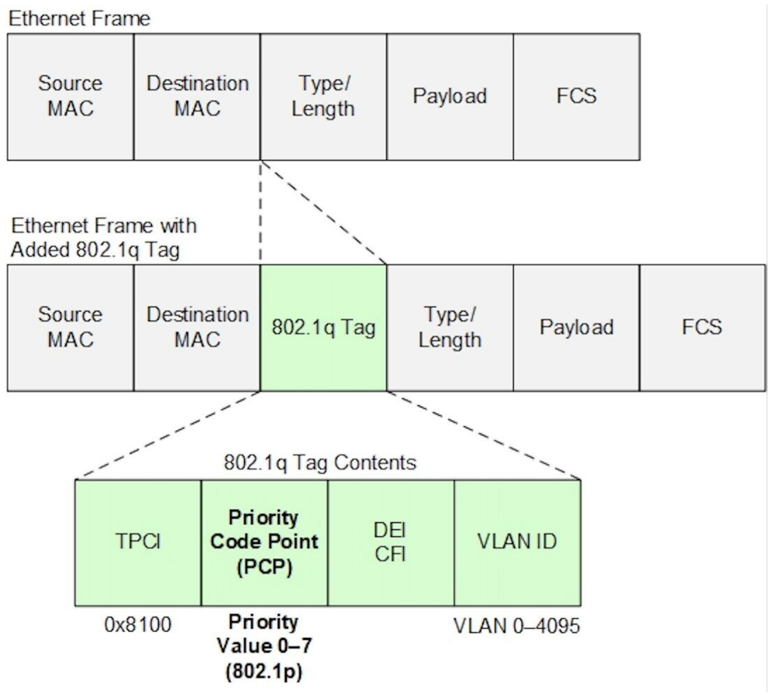
Feature Name	Release Information	Feature Description
QoS Support for VPWS	Release 7.7.1	Now, you can apply QoS packet classification on Layer 2 subinterfaces, and for VPWS traffic. This feature provides ingress classification support for L2 attachment circuit (AC) traffic based on the 802.1p field priority value. With this feature, you can use QoS policies on VPWS traffic in your network on Cisco 8000 Series routers.

EVPN-VPWS is a BGP control plane solution for point-to-point services. It implements the signaling and encapsulation techniques for establishing an EVPN instance between a pair of PEs. It has the ability to forward traffic from one network to another without a MAC lookup.

By configuring QoS, you can provide preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size. The device sends the packets without any assurance of reliability, delay bounds, or throughput.

The 802.1Q standard (and 802.1ad, a subset of 802.1Q) defines a system of VLAN tagging for Ethernet frames and also contains a provision for a QoS prioritization scheme known as 802.1p, which indicates the priority level of the frame, as shown in the figure. For more information on IEEE standards, browse the IEEE website.

Figure 2: 802.1p



VLAN Tag Priority Field Based Classification

The QoS term class of service (CoS) is a 3-bit field called Priority Code Point (PCP) which specifies a priority value between 0 and 7 that is used by QoS to differentiate traffic. Drop Eligible Indicator (DEI) is a 1-bit field that is used to indicate frames eligible to be dropped during traffic congestion.

MQC allows configuration of class-map match condition based on the PCP and DEI fields. The classification is supported on 802.1Q and 802.1ad interfaces.

Note The ingress classification supports AC-to-AC traffic flow and AC-to-PWE traffic flow.

Configuration Example for QoS Support for VPWS

Follow these steps to enable this feature:

- Enable VPWS configuration. Refer the L2VPN Configuration Guide for details.
- Configure ingress traffic classification, based on the PCP and DEI fields in the VLAN header.
  - Create class maps for different traffic classes.
  - Associate them with a policy map.
- Configure ingress traffic remarking.

Configure Ingress Traffic Classification Based on PCP and DEI Fields in the VLAN Header

```
/* Create class maps for different traffic classes */

Router# configure terminal
Router(config)# class-map match-all CONTROL_PLANE
Router(config-cmap)# match cos 7
Router(config-cmap)# end-class-map

Router(config)# class-map match-all VOIP
Router(config-cmap)# match cos 6
Router(config-cmap)# end-class-map

Router(config)# class-map match-all VIDEO_STREAM
Router(config-cmap)# match cos 5
Router(config-cmap)# end-class-map

Router(config)# class-map match-all TRANSACTIONAL_DATA
Router(config-cmap)# match cos 4
Router(config-cmap)# end-class-map

Router(config)# class-map match-all DB_SYNC
Router(config-cmap)# match cos 3
Router(config-cmap)# match dei 1
Router(config-cmap)# end-class-map

Router(config)# class-map match-all BULK_DATA
Router(config-cmap)# match cos 2
Router(config-cmap)# match dei 1
Router(config-cmap)# end-class-map

Router(config)# class-map match-all SCAVENGER
Router(config-cmap)# match cos 1
Router(config-cmap)# match dei 1
Router(config-cmap)# end-class-map
Router(config)# commit

/* Create a policy map and associate the class maps to it */

Router# configure terminal
Router(config)# policy-map INGRESS_L2_AC
Router(config-pmap)# class CONTROL_PLANE
Router(config-pmap-c)# set traffic-class 7
Router(config-pmap-c)# exit

Router(config-pmap)# class VOIP
Router(config-pmap-c)# set traffic-class 6
Router(config-pmap-c)# exit

Router(config-pmap)# class VIDEO_STREAM
Router(config-pmap-c)# set traffic-class 5
Router(config-pmap-c)# exit

Router(config-pmap)# class TRANSACTIONAL_DATA
Router(config-pmap-c)# set traffic-class 4
Router(config-pmap-c)# exit

Router(config-pmap)# class DB_SYNC
Router(config-pmap-c)# set traffic-class 3
Router(config-pmap-c)# exit

Router(config-pmap)# class BULK_DATA
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# exit

Router(config-pmap)# class SCAVENGER
Router(config-pmap-c)# set traffic-class 1
Router(config-pmap-c)# exit

Router(config-pmap)# class class-default
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
Router(config)# commit
```

Configure Ingress Traffic Remarking

```
/* Set CoS and DEI values for traffic classes, as needed */

Router# configure terminal
Router(config)# policy-map INGRESS_L2_AC
Router(config-pmap)# class CONTROL_PLANE
Router(config-pmap-c)# set traffic-class 7
Router(config-pmap-c)# exit

Router(config-pmap)# class VOIP
Router(config-pmap-c)# set traffic-class 6
Router(config-pmap-c)# set cos 7
Router(config-pmap-c)# exit

Router(config-pmap)# class VIDEO_STREAM
Router(config-pmap-c)# set traffic-class 5
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit

Router(config-pmap)# class TRANSACTIONAL_DATA
Router(config-pmap-c)# set traffic-class 4
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit
Router(config-pmap)# class DB_SYNC
Router(config-pmap-c)# set traffic-class 3
Router(config-pmap-c)# set dei 0
Router(config-pmap-c)# exit

Router(config-pmap)# class BULK_DATA
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# set cos 3
Router(config-pmap-c)# set dei 0
Router(config-pmap-c)# exit

Router(config-pmap)# class SCAVENGER
Router(config-pmap-c)# set traffic-class 1
Router(config-pmap-c)# exit

Router(config-pmap)# class class-default
Router(config-pmap-c)# set dei 1
Router(config-pmap-c)# end-policy-map
Router(config)# commit
```

/* Associate Policy-Map INGRESS_L2_AC With the Designated Subinterface */

Before you enable the subinterface, ensure that the parent interface state is up.

```
Router(config)# interface hundredGigE 0/11/0/31.102
Router(config-subif)# service-policy input INGRESS_L2_AC
Router(config-subif)# commit
```

Verification

Verify ingress QoS policy configuration. In the output, you can see that traffic is classified and transmitted for some categories.

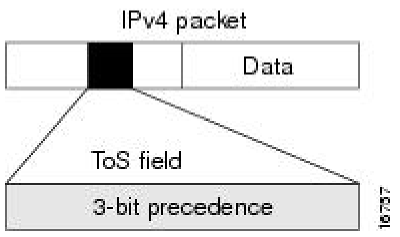
```
Router# show policy-map interface Hu0/11/0/31.102 input

HundredGigE0/11/0/31.102 input: INGRESS_L2_AC

Class CONTROL_PLANE
  Classification statistics          (packets/bytes)     (rate - kbps)
 Matched             :            9813350/13738690000          936847
    Transmitted         :            9813350/13738690000          936847
    Total Dropped       :                  0/0                    0
Class VOIP
  Classification statistics          (packets/bytes)     (rate - kbps)
 Matched             :          117760245/164864343000        11242157
    Transmitted         :          117760245/164864343000        11242157
    Total Dropped       :                  0/0                    0
Class VIDEO_STREAM
  Classification statistics          (packets/bytes)     (rate - kbps)
 Matched             :           49066792/68693508800         4684233
    Transmitted         :           49066792/68693508800         4684233
    Total Dropped       :                  0/0                    0
Class TRANSACTIONAL_DATA
  Classification statistics          (packets/bytes)     (rate - kbps)
 Matched             :          225707344/315990281600        21547467
    Transmitted         :          225707344/315990281600        21547467
    Total Dropped       :                  0/0                    0
Class DB_SYNC
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                    0
    Transmitted         :                  0/0                    0
    Total Dropped       :                  0/0                    0
```

```
Class BULK_DATA
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                    0
    Transmitted         :                  0/0                    0
    Total Dropped       :                  0/0                    0
Class SCAVENGER
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                    0
    Transmitted         :                  0/0                    0
    Total Dropped       :                  0/0                    0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
 Matched              :           500482413/700675378200         47779164
    Transmitted         :           500482413/700675378200         47779164
    Total Dropped       :                  0/0                    0
Policy Bag Stats time: 1657528790084  [Local Time: 07/11/22 08:39:50.084]
```

Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You can create differentiated service by setting precedence levels on incoming traffic and using them in combination with the QoS queuing features. So that, each subsequent network element can provide service based on the determined policy. IP precedence is usually deployed as close to the edge of the network or administrative domain as possible. This allows the rest of the core or backbone to implement QoS based on precedence.

Figure 3: IPv4 Packet Type of Service Field



You can use the three precedence bits in the type-of-service (ToS) field of the IPv4 header for this purpose. Using the ToS bits, you can define up to eight classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet in regard to the ToS to grant it. These other QoS features can assign appropriate traffic-handling policies, including congestion management strategy and bandwidth allocation. For example, queuing features such as LLQ can use the IP precedence setting of the packet to prioritize traffic.

IP Precedence Bits Used to Classify Packets

Use the three IP precedence bits in the ToS field of the IP header to specify the CoS assignment for each packet. You can partition traffic into a maximum of eight classes and then use policy maps to define network policies in terms of congestion handling and bandwidth allocation for each class.

Each precedence corresponds to a name. IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates. These names are defined in RFC 791.

IP Precedence Value Settings

By default, the routers leave the IP precedence value untouched. This preserves the precedence value set in the header and allows all internal network devices to provide service based on the IP precedence setting. This policy follows the standard approach stipulating that network traffic should be sorted into various types of service at the edge of the network and that those types of service should be implemented in the core of the network. Routers in the core of the network can then use the precedence bits to determine the order of transmission, the likelihood of packet drop, and so on.

Because traffic coming into your network can have the precedence set by outside devices, we recommend that you reset the precedence for all traffic entering your network. By controlling IP precedence settings, you prohibit users that have already set the IP precedence from acquiring better service for their traffic simply by setting a high precedence for all of their packets.

The class-based unconditional packet marking and LLQ features can use the IP precedence bits.

IP Precedence Compared to IP DSCP Marking

If you need to mark packets in your network and all your devices support IP DSCP marking, use the IP DSCP marking to mark your packets because the IP DSCP markings provide more unconditional packet marking options. If marking by IP DSCP is undesirable, however, or if you are unsure if the devices in your network support IP DSCP values, use the IP precedence value to mark your packets. The IP precedence value is likely to be supported by all devices in the network.

You can set up to 8 different IP precedence markings and 64 different IP DSCP markings.

Packet Classification on Your Router

On your router, there are two types of packet classification systems:

- In the ingress direction, QoS map and Ternary Content Addressable Memory (TCAM).
- In the egress direction, QoS map.

The router selects a QoS map-based classification system when you specify a QoS policy for matching on:

- Differentiated Services Code Point (DSCP) or precedence value (also called DSCP or Precedence-based classification)
- Experimental bits (EXP)
- Priority Code Point (PCP)
- Drop Eligibility Indicator (DEI), or
- qos-group.

Otherwise, the router selects TCAM as the packet classification system.

The TCAM is an extension of the Content Addressable Memory (CAM) table concept. A CAM table takes in an index or key value (usually a MAC address) and looks up the resulting value (usually a switch port or VLAN ID). Table lookup is fast and always based on an exact key match consisting of two input values: 0 and 1 bits.

The QoS map is a table-based classification system for traffic packets.

Classify and Remark Layer 3 Header on Layer 2 Interfaces

When you need to mark packets for Layer 2 interface traffic that flows across bridge domains and bridge virtual interfaces (BVIs), you can create a mixed QoS policy. This policy has both map-based and TCAM-based classification class-maps. The mixed policy ensures that both bridged (Layer 2) and Bridge Virtual Interface (BVI, or Layer 3) traffic flows are classified and remarked.

Guidelines

- A class-map with TCAM classification may not match bridged traffic. TCAM entries match only routed traffic while map entries match both bridged and BVI traffic.
- A class-map with map-based classification matches both bridged and BVI traffic.

Example

```
ipv4 access-list acl_v4
10 permit ipv4 host 100.1.1.2 any
20 permit ipv4 host 100.1.100.2 any
ipv6 access-list acl_v6
10 permit tcp host 50:1:1::2 any
20 permit tcp any host 50:1:200::2
class-map match-any c_match_acl
match access-group ipv4 acl_v4 ! This entry does not match bridged traffic
match access-group ipv6 acl_v6 ! This entry does not match bridged traffic
match dscp af11 This entry matches bridged and BVI traffic
class-map match-all c_match_all
match protocol udp ! This entry does not match bridged traffic
match prec 7
class-map match-any c_match_protocol
match protocol tcp ! This entry, and hence this class does not match bridged traffic
class-map match-any c_match_ef
match dscp ef ! This entry/class matches bridged and BVI traffic
class-map match-any c_qosgroup_1 This class matches bridged and BVI traffic
!
match qos-group 1
policy-map p_ingress
class c_match_acl
set traffic-class 1
set qos-group 1
!
class c_match_all
set traffic-class 2
set qos-group 2
!
class c_match_ef
set traffic-class 3
set qos-group 3
!
class c_match_protocol
set traffic-class 4
set qos-group 4
policy-map p_egress
class c_qosgroup_1
set dscp af23
interface FourHundredGigE0/0/0/0
l2transport
```

```
service-policy input p_ingress
service-policy output p_egress
!
!
interface FourHundredGigE0/0/0/1
ipv4 address 200.1.2.1 255.255.255.0
ipv6 address 2001:2:2::1/64
service-policy input p_ingress
service-policy output p_egress
```

Traffic Class Elements

The purpose of a traffic class is to classify traffic on your router. Use the **class-map** command to define a traffic class.

A traffic class contains three major elements:

- A name
- A series of **match** commands - to specify various criteria for classifying packets.
- An instruction on how to evaluate these **match** commands (if more than one **match** command exists in the traffic class)

Packets are checked to determine whether they match the criteria specified in the **match** commands. If a packet matches the specified criteria, that packet is considered a member of the class and is forwarded according to the QoS specifications set in the traffic policy. Packets that fail to meet any of the matching criteria are classified as members of the default traffic class.

This table shows the details of match types supported on the router.

Match Type Supported	Min, Max	Max Entries	Support for Match NOT	Support for Ranges	Direction Supported on Interfaces
IPv4 DSCP IPv6 DSCP	(0,63)	64	Yes	Yes	Ingress
DSCP					Egress
IPv4 Precedence IPv6 Precedence	(0,7)	8	Yes	No	Ingress
Precedence					Egress
MPLS Experimental Topmost	(0,7)	8	Yes	No	Ingress Egress
Access-group	Not applicable	8	No	Not applicable	Ingress
Match qos-group	(1-31)	7 + class-default	No	No	Egress

Match Type Supported	Min, Max	Max Entries	Support for Match NOT	Support for Ranges	Direction Supported on Interfaces
Protocol	(0, 255)	1	Yes	Not applicable	Ingress
CoS	(0,7)	8	Yes	No	Ingress and Egress
DEI	(0,1)	2	Yes	No	Ingress and Egress

Default Traffic Class

Unclassified traffic (traffic that doesn't meet the match criteria specified in the traffic classes) is treated as belonging to the default traffic class.

If the user doesn't configure a default class, packets are still treated as members of the default class. However, by default, the default class has no enabled features. Therefore, packets belonging to a default class with no configured features have no QoS functionality.

For egress classification, match on **qos-group** for seven groups with range (1-31) is supported. Match **qos-group 0** can't be configured. The class-default in the egress policy maps to **qos-group 0**.

This example shows how to configure a traffic policy for the default class:

```
configure
 policy-map ingress_policy1
 class class-default
  police rate percent 30
 !
```

Create a Traffic Class

To create a traffic class containing match criteria, use the **class-map** command to specify the traffic class name, and then use the **match** commands in class-map configuration mode, as needed.

Guidelines

- Users can provide multiple values for a match type in a single line of configuration; that is, if the first value does not meet the match criteria, then the next value indicated in the match statement is considered for classification.
- Use the **not** keyword with the **match** command to perform a match based on the values of a field that are not specified.
- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify **match-all**, the traffic must match all the match criteria.

- For the **match access-group** command, QoS classification based on the packet length or TTL (time to live) field in the IPv4 and IPv6 headers is not supported.
- For the **match access-group** command, when an ACL list is used within a class-map, the deny action of the ACL is ignored and the traffic is classified based on the specified ACL match parameters.
- The **match qos-group**, **traffic-class**, **DSCP/Prec**, and **MPLS EXP** are supported only in egress direction, and these are the only match criteria supported in egress direction
- The egress default class implicitly matches **qos-group** 0.
- Multicast takes a system path that is different than unicast on router, and they meet later on the egress in a multicast-to-unicast ratio of 20:80 on a per interface basis. This ratio is maintained on the same priority level as that of the traffic.
- Egress QoS for multicast traffic treats traffic classes 0-5 as low-priority and traffic classes 6-7 as high priority. Currently, this is not user-configurable.
- Egress shaping does not take effect for multicast traffic in the high priority (HP) traffic classes. It only applies to unicast traffic.
- If you set a traffic class at the ingress policy and do not have a matching class at egress for the corresponding traffic class value, then the traffic at ingress with this class will not be accounted for in the default class at the egress policy map.
- Only traffic class 0 falls in the default class. A non-zero traffic class assigned on ingress but with no assigned egress queue, falls neither in the default class nor any other class.

Configuration Example

You have to accomplish the following to complete the traffic class configuration:

1. Creating a class map
2. Specifying the match criteria for classifying the packet as a member of that particular class

   (For a list of supported match types, see Traffic Class Elements, on page 17.)

```
Router# configure
Router(config)# class-map match-any qos-1
Router(config-cmap)# match qos-group 1
Router(config-cmap)# end-class-map
Router(config-cmap)# commit
```

Use this command to verify the class-map configuration:

```
Router#show class-map qos-1
1) ClassMap: qos-1    Type: qos
    Referenced by 2 Policymaps
```

Also see, Attach a Traffic Policy to an Interface, on page 21.

Related Topics

- Traffic Class Elements, on page 17

- Traffic Policy Elements, on page 20

Traffic Policy Elements

A traffic policy contains three elements:

- Name
- Traffic class
- QoS policies

After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to be applied to the classified traffic.

The MQC does not necessarily require that the users associate only one traffic class to one traffic policy.

The order in which classes are configured in a policy map is important. The match rules of the classes are programmed into the TCAM in the order in which the classes are specified in a policy map. Therefore, if a packet can possibly match multiple classes, only the first matching class is returned and the corresponding policy is applied.

The router supports 8 classes per policy-map in the ingress direction and 8 classes per policy-map in the egress direction.

This table shows the supported class-actions on the router.

Supported Action Types	Direction supported on Interfaces
bandwidth-remaining	egress
mark	See Packet Marking, on page 26
police	ingress
priority	egress (level 1 to level 7)
queue-limit	egress
shape	egress
red	egress

RED supports the **discard-class** option; the only values to be passed to the discard-class being 0 and 1.

Create a Traffic Policy

The purpose of a traffic policy is to configure the QoS features that should be associated with the traffic that has been classified in a user-specified traffic class or classes.

To configure a traffic class, see Create a Traffic Class, on page 18.

After you define a traffic policy with the **policy-map** command, you can attach it to one or more interfaces to specify the traffic policy for those interfaces by using the **service-policy** command in interface configuration mode. With dual policy support, you can have two traffic policies, one marking and one queuing attached at the output. See, Attach a Traffic Policy to an Interface, on page 21.

Note If multicast route statistics (accounting) is enabled and if an input QoS policy is applied to incoming multicast traffic on an interface, QoS classification counters may not update for the input multicast traffic. This does not affect any of the QoS functionalities for multicast traffic.

To update QoS counters for the incoming multicast traffic, you must disable the accounting for multicast routing by running the **no accounting per-prefix** command.

For details about the **no accounting per-prefix** command, see the *Multicast Command Reference for Cisco 8000 Series Routers*.

Configuration Example

You have to accomplish the following to complete the traffic policy configuration:

1. Creating 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 class-action(s) (see Traffic Policy Elements, on page 20)

```
Router# configure
Router(config)# policy-map  test-shape-1
Router(config-pmap)# class qos-1

/* Configure class-action ('shape' in this example).
Repeat as required, to specify other class-actions */
Router(config-pmap-c)# shape average percent 40
Router(config-pmap-c)# exit

/* Repeat class configuration as required, to specify other classes */

Router(config-pmap)# end-policy-map
Router(config)# commit
```

Related Topics

- Traffic Policy Elements, on page 20
- Traffic Class Elements, on page 17

Attach a Traffic Policy to an Interface

After the traffic class and the traffic policy are created, you must attach the traffic policy to interface, and specify the direction in which the policy should be applied.

Note Hierarchical policies are not supported.

When a policy-map is applied to an interface, the transmission rate counter of each class is not accurate. This is because the transmission rate counter is calculated based on the exponential decay filter.

Configuration Example

You have to accomplish the following to attach a traffic policy to an interface:

1. Creating a traffic class and the associated rules that match packets to the class (see Create a Traffic Class, on page 18)
2. Creating a traffic policy that can be attached to one or more interfaces to specify a service policy (see Create a Traffic Policy, on page 20)
3. Associating the traffic class with the traffic policy
4. Attaching the traffic policy to an interface, in the ingress or egress direction

```
Router# configure
Router(config)#  interface fourHundredGigE 0/0/0/2
Router(config-int)# service-policy output strict-priority
Router(config-int)# commit
```

Running Configuration

```
/* Class-map configuration */

class-map match-any traffic-class-7
 match traffic-class 7
 end-class-map

!class-map match-any traffic-class-6
 match traffic-class 6
 end-class-map

class-map match-any traffic-class-5
 match traffic-class 5
 end-class-map

class-map match-any traffic-class-4
 match traffic-class 4
 end-class-map

class-map match-any traffic-class-3
 match traffic-class 3

class-map match-any traffic-class-2
 match traffic-class 2
 end-class-map

class-map match-any traffic-class-1
 match traffic-class 1
 end-class-map
```

```
/* Traffic policy configuration */

policy-map test-shape-1
 class traffic-class-1
  shape average percent 40
 !

policy-map strict-priority
 class tc7
  priority level 1
  queue-limit 75 mbytes
 !
 class tc6
  priority level 2
  queue-limit 75 mbytes
 !
 class tc5
  priority level 3
  queue-limit 75 mbytes
 !
 class tc4
  priority level 4
  queue-limit 75 mbytes
 !
 class tc3
  priority level 5
  queue-limit 75 mbytes
 !
 class tc2
  priority level 6
  queue-limit 75 mbytes
 !
 class tc1
  priority level 7
  queue-limit 75 mbytes
 !
 class class-default
  queue-limit 75 mbytes
 !
 end-policy-map

- - -
- - -

/* Attaching traffic policy to an interface in egress direction */
interface fourHundredGigE 0/0/0/2
 service-policy output strict-priority
 !
```

Verification

```
Router# #show qos int fourHundredGigE 0/0/0/2 output

NOTE:- Configured values are displayed within parentheses Interface FourHundredGigE0/0/0/2
 ifh 0xf0001c0  -- output policy

NPU Id:                        0
Total number of classes:       8
Interface Bandwidth:           400000000 kbps
Policy Name:                   strict-priority
```

```
VOQ Base:                      2400
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class (HP1)                        =   tc7
Egressq Queue ID                          =   2407 (HP1 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP2)                        =   tc6
Egressq Queue ID                          =   2406 (HP2 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP3)                        =   tc5
Egressq Queue ID                          =   2405 (HP3 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP4)                        =   tc4
Egressq Queue ID                          =   2404 (HP4 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP5)                        =   tc3
Egressq Queue ID                          =   2403 (HP5 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP6)                        =   tc2
Egressq Queue ID                          =   2402 (HP6 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP7)                        =   tc1
Egressq Queue ID                          =   2401 (HP7 queue)
Queue Max. BW.                            =   no max (default)
TailDrop Threshold                        =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class                              =   class-default
Egressq Queue ID                          =   2400 (Default LP queue)
Queue Max. BW.                            =   no max (default)
Inverse Weight / Weight                   =   1 / (BWR not configured)
TailDrop Threshold                        =   74999808 bytes / 150 ms (75 megabytes)
WRED not configured for this class

!
```

Related Topics

- Traffic Policy Elements, on page 20
- Traffic Class Elements, on page 17

CHAPTER 4

Mark Packets to Change Priority Settings

Packet Marking Overview

You can use packet marking in input policy maps to set or modify the attributes for traffic belonging to a specific class. For example, you can change the CoS value in a class or set IP DSCP or IP precedence values for a specific type of traffic. These new values are then used to determine how the traffic should be treated.

Default Marking

When an ingress or egress interface adds VLAN tags or MPLS labels, it requires a default value for the class of service and EXP values that go into those tags and labels.

On the router, one ingress default QoS mapping profile and one egress default QoS mapping profile are created and configured per device during initialization.

QoS Behavior for Generic Routing Encapsulation (GRE) Tunnels

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
QoS Behavior for Generic Routing Encapsulation (GRE) Tunnels: Default Marking	Release 7.3.1	With the support for GRE encapsulation and decapsulation tunnel interfaces, there are some important updates to QoS behavior for GRE tunnels. These updates are applicable for default packet marking and involve Type of Service (ToS) and MPLS experimental bits.

GRE Encapsulation

If you do not configure Type of Service (ToS), the outer IP precedence value or the differentiated services code point (DSCP) value is copied from the inner IP header. If you configure ToS, the outer IP precedence value or DCSP value is as per the ToS configuration.

GRE Decapsulation

During decapsulation, the MPLS experimental bits (EXP) are derived from the outer IP packet.

For more information on GRE tunnels, see the *Interfaces Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.3.x*.

Packet Marking

The packet marking feature, also called explicit marking, provides users with a means to differentiate packets based on the designated markings. The router supports ingress and egress packet marking.

Supported Packet Marking Operations

This table shows the supported packet marking operations.

Table 6: Supported Packet Marking Operations for Egress and Ingress

Supported Mark Types	Range	Support for Unconditional Marking	Support for Conditional Marking
set discard-class	0-1	ingress	No
set dscp	0-63	ingress, egress	No
set mpls experimental topmost	0-7	ingress, egress	No
set precedence	0-7	ingress, egress	No
set qos-group	0-31	ingress	No
set cos	0-7	ingress, egress	No
set dei	0-1	ingress, egress	No

QoS Behavior for Generic Routing Encapsulation (GRE) Tunnels

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
QoS Behavior for Generic Routing Encapsulation (GRE) Tunnels: Explicit Marking	Release 7.3.1	With the support for GRE encapsulation and decapsulation tunnel interfaces, there are some important updates to QoS behavior for GRE tunnels. These updates are applicable for explicit packet marking and involve QoS behavior during ingress and egress.

GRE Encapsulation

During encapsulation of IPv4/IPv6 payload inside the GRE header, QoS behavior is as follows:

- Ingress: QoS supports classification on the payload Layer 3 fields or EXP and remarking payload IP header DSCP.
- Egress: QoS supports setting outer GRE IP header DSCP. It doesn’t overwrite the Tunnel Type of Service (ToS) configuration and doesn’t remark GRE IP header DCSP.

GRE Decapsulation

During decapsulation of the outer GRE header (during which the inner IPv4/IPv6/MPLS payload is forwarded to the next-hop router), QoS behavior is as follows:

- Ingress: QoS supports classification on Layer 3 fields of outer GRE using the **set qos-group** command. Setting DSCP on the ingress interface sets DSCP for the inner headers.
- Egress: QoS supports classification using **qos-group** to set DSCP or EXP for egress packets.

For more information on GRE tunnels, see the *Interfaces Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.3.x*.

Classification and Marking for L2 Traffic

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Classification and Marking for L2 Traffic	Release 7.7.1	You can now perform packet remarking on Layer 2 subinterfaces. Traffic marking allows you to fine tune the traffic attributes on your network. The increased granularity helps single out traffic that requires special handling, and achieves optimal application performance.

Traffic marking is a method used to identify certain traffic types for unique handling, effectively partitioning network traffic into different categories. Cisco 8000 Series routers support Layer 2 marking of Ethernet packets in the egress and ingress directions.

Configuration Example for Classification and Marking for L2 Traffic

Follow these steps to enable this feature:

For guidelines on creating a traffic class and egress traffic classification, refer the *Create a Traffic Class* section in the *Classify Packets to Identify Specific Traffic* chapter.

- Enable Layer 2 configuration. Refer the L2VPN Configuration Guide for details.
- Enable egress remarking with the **qos-group** match type.
- Verify the egress remark QoS policy configuration.
- Enable egress traffic remarking.

Egress Remarking With *qos-group* Match Type

```
/* Associate a specific qos-group with a traffic class using the set qos-group command */

Router# configure terminal
Router(config)# policy-map INGRESS_L2_AC
Router(config-pmap)# class CONTROL_PLANE
Router(config-pmap-c)# set traffic-class 7
Router(config-pmap-c)# set qos-group 27
Router(config-pmap-c)# exit

Router(config-pmap)# class VOIP
Router(config-pmap-c)# set traffic-class 6
Router(config-pmap-c)# set qos-group 26
Router(config-pmap-c)# exit

Router(config-pmap)# class VIDEO_STREAM
Router(config-pmap-c)# set traffic-class 5
Router(config-pmap-c)# set qos-group 25
Router(config-pmap-c)# exit

Router(config-pmap)# class TRANSACTIONAL_DATA
Router(config-pmap-c)# set traffic-class 4
Router(config-pmap-c)# set qos-group 24
Router(config-pmap-c)# exit

Router(config-pmap)# class DB_SYNC
Router(config-pmap-c)# set traffic-class 3
Router(config-pmap-c)# set qos-group 23
Router(config-pmap-c)# exit

Router(config-pmap)# class BULK_DATA
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# set qos-group 22
Router(config-pmap-c)# exit

Router(config-pmap)# class SCAVENGER
Router(config-pmap-c)# set traffic-class 1
Router(config-pmap-c)# set qos-group 21
Router(config-pmap-c)# exit

Router(config-pmap)# class class-default
Router(config-pmap-c)# end-policy-map
Router(config)# commit
```

```
/* Associate egress policy map with the qos-groups  */

Router# configure terminal
Router(config)# policy-map REMARK_EGRESS_QG_L2_AC
Router(config-pmap)# class QoS_GROUP_27
Router(config-pmap-c)# set cos 7
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_26
Router(config-pmap-c)# set cos 7
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_25
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_24
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_23
Router(config-pmap-c)# set cos 3
Router(config-pmap-c)# set dei 0
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_22
Router(config-pmap-c)# set cos 3
Router(config-pmap-c)# set dei 1
Router(config-pmap-c)# exit

Router(config-pmap)# class QoS_GROUP_21
Router(config-pmap-c)# set cos 1
Router(config-pmap-c)# set dei 1
Router(config-pmap-c)# exit

Router(config-pmap)# class class-default
Router(config-pmap-c)# set dei 1
Router(config-pmap-c)# exit
Router(config)# commit
```

/* Associate Policy-Map REMARK_EGRESS_QG_L2_AC With the Designated Subinterface */

```
Router(config)# interface bundle-Ether 1.104
Router(config-subif)# service-policy output REMARK_EGRESS_QG_L2_AC
Router(config-subif)# commit
```

Verification

Verify the egress remark QoS policy configuration:

```
Router# show policy-map interface Bundle-Ether 1.104 output

Bundle-Ether1.104 output: REMARK_EGRESS_QG_L2_AC

Class QoS_GROUP_7
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           21319554/29847375600          985424
    Transmitted         :           21319554/29847375600          985424
    Total Dropped       :                  0/0                    0
Class QoS_GROUP_6
```

```
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          21581741/30214437400          997540
    Transmitted         :          21581741/30214437400          997540
    Total Dropped       :                 0/0                    0
Class QoS_GROUP_5
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         106597824/149236953600         4927114
    Transmitted         :         106597824/149236953600         4927114
    Total Dropped       :                 0/0                    0
Class QoS_GROUP_4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         490350115/686490161000         22664721
    Transmitted         :         490350115/686490161000         22664721
    Total Dropped       :                 0/0                    0
Class QoS_GROUP_3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                 0/0                    0
    Transmitted         :                 0/0                    0
    Total Dropped       :                 0/0                    0
Class QoS_GROUP_2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                 0/0                    0
    Transmitted         :                 0/0                    0
    Total Dropped       :                 0/0                    0
Class QoS_GROUP_1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                 0/0                    0
    Transmitted         :                 0/0                    0
    Total Dropped       :                 0/0                    0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :        1087299229/1522218920600        50256530
    Transmitted         :        1087299229/1522218920600        50256530
    Total Dropped       :                 0/0                    0
Policy Bag Stats time: 1657531125148  [Local Time: 07/11/22 09:18:45.148]
```

Remarking of Egress Traffic

```
Router# configure terminal
Router(config)# policy-map REMARK_EGRESS_L2_AC
Router(config-pmap)# class CONTROL_PLANE
Router(config-pmap-c)# set traffic-class 7
Router(config-pmap-c)# exit

Router(config-pmap)# class VOIP
Router(config-pmap-c)# set traffic-class 6
Router(config-pmap-c)# set cos 7
Router(config-pmap-c)# exit

Router(config-pmap)# class VIDEO_STREAM
Router(config-pmap-c)# set traffic-class 5
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit

Router(config-pmap)# class TRANSACTIONAL_DATA
Router(config-pmap-c)# set traffic-class 4
Router(config-pmap-c)# set cos 5
Router(config-pmap-c)# exit

Router(config-pmap)# class DB_SYNC
Router(config-pmap-c)# set traffic-class 3
Router(config-pmap-c)# set dei 0
Router(config-pmap-c)# exit
Router(config-pmap)# class BULK_DATA
Router(config-pmap-c)# set traffic-class 2
Router(config-pmap-c)# set cos 3
Router(config-pmap-c)# set dei 0
Router(config-pmap-c)# exit

Router(config-pmap)# class SCAVENGER
Router(config-pmap-c)# set traffic-class 1
Router(config-pmap-c)# exit

Router(config-pmap)# class class-default
Router(config-pmap-c)# set dei 1
Router(config-pmap-c)# end-policy-map
Router(config)# commit
```

Class-based Unconditional Packet Marking Feature and Benefits

The packet marking feature allows you to partition your network into multiple priority levels or classes of service, as follows:

- Use QoS unconditional packet marking to set the IP precedence or IP DSCP values for packets entering the network. Routers within your network can then use the newly marked IP precedence values to determine how the traffic should be treated.

  On ingress direction, after matching the traffic based on either the IP Precedence or DSCP value, you can set it to a particular discard-class. Weighted random early detection (WRED), a congestion avoidance technique, thereby uses discard-class values to determine the probability that a packet is dropped.

- Use QoS unconditional packet marking to assign MPLS packets to a QoS group. The router uses the QoS group to determine how to prioritize packets for transmission. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map class configuration mode.

Note Setting the QoS group identifier does not automatically prioritize the packets for transmission. You must first configure an egress policy that uses the QoS group.

- Mark Multiprotocol Label Switching (MPLS) packets by setting the EXP bits within the imposed or topmost label.
- Mark packets by setting the value of the *qos-group* argument.
- Mark packets by setting the value of the *discard-class* argument.

Note *qos-group* and *discard-class* are variables internal to the router, and are not transmitted.

The configuration task is described in the Configure Class-based Unconditional Packet Marking, on page 32.

Configure Class-based Unconditional Packet Marking

This configuration task explains how to configure the following class-based, unconditional packet marking features on your router:

- IP precedence value
- IP DSCP value
- QoS group value (ingress only)
- CoS value
- MPLS experimental value
- Discard class

Note IPv4 and IPv6 QoS actions applied to MPLS tagged packets are not supported. The configuration is accepted, but no action is taken.

Configuration Example

Follow these steps to configure unconditional packet marking features on your router.

1. Create or modify a policy map that can be attached to one or more interfaces to specify a service policy and enter the policy map configuration mode.
2. Configure an interface and enter the interface configuration mode.
3. Attach a policy map to an input or output interface to be used as the service policy for that interface.

Configuration Example

```
router# configure
router(config)# interface hundredGigE 0/0/0/24
router(config-pmap)# policy-map policy1
Router(config-int)# commit
```

Running Configuration

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

class-map match-any class1
 match protocol ipv4
 end-class-map
!
!
policy-map policy1
 class class1
  set precedence 1
 !
 class class-default
 !
 end-policy-map
!
interface HundredGigE0/0/0/24
 service-policy input policy1

!
```

Verification

Run this command to display policy configuration information for all classes configured for all service policies on the specified interface.

```
router# show run interface hundredGigE 0/0/0/24
```

Class-based Unconditional Packet Marking: Examples

These are typical examples for class-based unconditional packet marking.

IP Precedence Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a previously defined class map called *class1* through the use of the **class** command, and then the service policy is attached to the output HundredGigE interface 0/7/0/1. The IP precedence bit in the ToS byte is set to 1:

```
policy-map policy1
  class class1
    set precedence 1
!
interface HundredGigE 0/7/0/1
  service-policy output policy1
```

IP DSCP Marking Configuration: Example

In this example, a service policy called policy1 is created. This service policy is associated to a previously defined class map through the use of the **class** command. In this example, it is assumed that a class map called class1 was previously configured and new class map called class2 is created.

In this example, the IP DSCP value in the ToS byte is set to 5:

```
policy-map policy1
  class class1
    set dscp 5

  class class2
    set dscp ef
```

After you configure the settings shown for voice packets at the edge, all intermediate routers are configured to provide low-latency treatment to the voice packets, as follows:

```
class-map voice
  match dscp ef
```

```
policy-map qos-policy
  class voice
    priority level 1
    police rate percent 10
```

QoS Group Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a HundredGigE 0/7/0/1. The qos-group value is set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set qos-group 1
  !
interface HundredGigE 0/7/0/1
  service-policy input policy1
```

Note The **set qos-group** command is supported only on an ingress policy.

CoS Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a HundredGigE 0/7/0/1.100. The IEEE 802.1p (CoS) bits in the Layer 2 header are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set cos 1
  !

interface HundredGigE 0/7/0/1.100
  service-policy input policy1
```

MPLS Experimental Bit Imposition Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a HundredGigE 0/7/0/1. The MPLS EXP bits of all imposed labels are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101
```

```
policy-map policy1
  class class1
    set mpls exp imposition 1
 !
interface HundredGigE 0/7/0/1
  service-policy input policy1
```

Note The **set mpls exp imposition** command is supported only on an ingress policy.

MPLS Experimental Topmost Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a HundredGigE 0/7/0/1. The MPLS EXP bits on the TOPMOST label are set to 1:

```
class-map match-any class1
  match mpls exp topmost 2

policy-map policy1
  class class1
    set mpls exp topmost 1
  !
interface HundredGigE 0/7/0/1
  service-policy output policy1
```

Queuing and Marking Policies: Examples

These examples explain how queuing and marking can be applied to an interface.

Ingress Traffic Classification and QoS Mapping Configuration for Input Interfaces

In this example, a policy map named "INGRESS_L3" is configured, specifically for classifying and marking incoming traffic.

```
RP/0/RP0/CPU0:Router#show policy-map pmap-name INGRESS_L3 detail

class-map match-all CONTROL_PLANE
match dscp cs7
 end-class-map
!
class-map match-all VOIP
match dscp cs6
 end-class-map
!
class-map match-all VIDEO_STREAM
match dscp cs5
 end-class-map
!
class-map match-all TRANSACTIONAL_DATA
match dscp cs4
 end-class-map
!
class-map match-all DB_SYNC
match dscp cs3
 end-class-map
!
```

```
class-map match-all BULK_DATA
match dscp cs2
 end-class-map
!
class-map match-all SCAVENGER
match dscp cs1
 end-class-map
!
policy-map INGRESS_L3
class CONTROL_PLANE
  set traffic-class 7
  set qos-group 27
!
 class VOIP
  set traffic-class 6
  set qos-group 26
!
 class VIDEO_STREAM
  set traffic-class 5
  set qos-group 25
!
 class TRANSACTIONAL_DATA
  set traffic-class 4
  set qos-group 24
!
 class DB_SYNC
  set traffic-class 3
  set qos-group 23
!
 class BULK_DATA
  set traffic-class 2
  set qos-group 22
!
 class SCAVENGER
  set traffic-class 1
  set qos-group 21
!
 class class-default
!
 end-policy-map
!
```

Egress Queueing Policy Configuration

In this example, a policy map named "EGRESS_QOS_QUEUEING" is configured for the egress (outgoing) direction.

```
RP/0/RP0/CPU0:Router#show policy-map pmap-name EGRESS_QOS_QUEUEING detail

class-map match-any TC7_CONTROL_PLANE
match traffic-class 7
 end-class-map
!
class-map match-any TC6_VOIP
match traffic-class 6
 end-class-map
!
class-map match-any TC5_VIDEO_STREAM
match traffic-class 5
 end-class-map
!
class-map match-any TC4_TRANSACTIONAL_DATA
match traffic-class 4
 end-class-map
```

```
!
class-map match-any TC3_DB_SYNC
match traffic-class 3
 end-class-map
!
class-map match-any TC2_BULK_DATA
match traffic-class 2
 end-class-map
!
class-map match-any TC1_SCAVENGER
match traffic-class 1
 end-class-map
!
policy-map EGRESS_QOS_QUEUEING
class TC7_CONTROL_PLANE
  priority level 1
 !
 class TC6_VOIP
  priority level 2
 !
 class TC5_VIDEO_STREAM
  bandwidth remaining ratio 20
  random-detect 50 mbytes 100 mbytes
 !
 class TC4_TRANSACTIONAL_DATA
  bandwidth remaining ratio 20
  queue-limit 75 mbytes
 !
 class TC3_DB_SYNC
  bandwidth remaining ratio 10
 !
 class TC2_BULK_DATA
  bandwidth remaining ratio 10
 !
 class TC1_SCAVENGER
  bandwidth remaining ratio 5
 !
 class class-default
!
 end-policy-map
!
```

Egress Marking Policy Configuration

In this example, a policy map named "EGRESS_REMARK" is configured for the egress (outgoing) direction.

```
RP/0/RP0/CPU0:Router#show policy-map pmap-name EGRESS_REMARK detail

class-map match-any QG4
match qos-group 4
 end-class-map
!
class-map match-any QG3
match qos-group 3
 end-class-map
!
class-map match-any QG2
match qos-group 2
 end-class-map
!
policy-map EGRESS_REMARK
class QG4
  set dscp af42
!
```

```
 class QG3
  set dscp af33
!
 class QG2
  set dscp af21
!
 class class-default
!
 end-policy-map
!
```

Output Interface Configuration with Queueing Policy and Remark Policy Applied

```
interface FourHundredGigE0/2/0/29
mtu 10000
service-policy output EGRESS_REMARK
service-policy output EGRESS_QOS_QUEUEING
ipv4 address 18.29.0.1 255.255.255.0
ipv6 address 18:1d::1/96
!
```

IP Precedence Compared to IP DSCP Marking

If you need to mark packets in your network and all your devices support IP DSCP marking, use the IP DSCP marking to mark your packets because the IP DSCP markings provide more unconditional packet marking options. If marking by IP DSCP is undesirable, however, or if you are unsure if the devices in your network support IP DSCP values, use the IP precedence value to mark your packets. The IP precedence value is likely to be supported by all devices in the network.

You can set up to 8 different IP precedence markings and 64 different IP DSCP markings.

Configure DSCP CS7 (Precedence 7)

See the following example to configure options in DSCP for a particular source address in IPv4 packets.

Configuration Example

```
policy-map policy1
 class class1
  set dscp cs7
 !
```

In-Place Policy Modification

The In-Place policy modification feature allows you to modify a QoS policy even when the QoS policy is attached to one or more interfaces. A modified policy is subjected to the same checks that a new policy is subject to when it is bound to an interface. If the policy-modification is successful, the modified policy takes effect on all the interfaces to which the policy is attached. However, if the policy modification fails on any one of the interfaces, an automatic rollback is initiated to ensure that the pre-modification policy is in effect on all the interfaces.

You can also modify any class map used in the policy map. The changes made to the class map take effect on all the interfaces to which the policy is attached.

Note

- The QoS statistics for the policy that is attached to an interface are lost (reset to 0) when the policy is modified.
- When a QoS policy attached to an interface is modified, there might not be any policy in effect on the interfaces in which the modified policy is used for a short period of time.
- An in-place modification of an ACL does not reset the policy-map statistics counter.

Verification

If unrecoverable errors occur during in-place policy modification, the policy is put into an inconsistent state on target interfaces. No new configuration is possible until the configuration session is unblocked. It is recommended to remove the policy from the interface, check the modified policy and then re-apply accordingly.

Recommendations for Using In-Place Policy Modification

For a short period of time while a QoS policy is being modified, there might not be any policy in effect on the interfaces in which the modified policy is used. For this reason, modify QoS policies that affect the fewest number of interfaces at a time. Use the **show policy-map targets** command to identify the number of interfaces that will be affected during policy map modification.

CHAPTER 5

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, configure the **hw-module profile qos voq-mode** command and 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.

Subinterface Queueing Policy

Each subinterface 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 subinterface, apply a queuing policy on that subinterface. When you remove the subinterface queuing policy, the associated VoQs are freed and the subinterface 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.

Virtual Output Queue Watchdog

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Output Queue Watchdog	Release 7.3.6	You can now quickly detect any stuck Virtual Output Queues (VOQ) in a line or fabric card within a minute. If the router detects a stuck queue on a line card, it automatically shuts down the line card to prevent potential issues. If the router detects a stuck queue on a fabric card, it triggers a hard reset on the Network Processing Unit (NPU). This ensures the continuous movement of traffic queues, which is crucial for enforcing QoS policies, even when hardware issues disrupt the VOQ and impede the flow of traffic. The feature is enabled by default and can be disabled using the command **hw-module voq-watchdog feature disable**. The feature is supported only in Cisco 8000 Series Routers with Cisco Silicon One Q200 ASICs. The feature introduces these changes: **CLI:** • **hw-module voq-watchdog feature disable** • **hw-module voq-watchdog cardshut disable**

Virtual Output Queue (VOQ) is a mechanism to manage a situation where a packet at the front of the queue is unable to move forward due to a busy destination, causing a delay for all packets behind it, even those with available output ports. VOQ prevents packet delay or loss by providing each output port with a dedicated queue for every input port. This mechanism ensures that packets are transmitted in the correct order and maintains quality of service (QoS) by managing congestion.

A VOQ can become stuck when it fails to transmit traffic for a certain period despite the non-empty queue. There are a few scenarios in which this can happen. For example, when a port or traffic class (Output Queue, OQ) is PFC-paused (Priority Flow Control) by the peer, all the VOQs sending traffic to this OQ are also paused. Another scenario is when a port's higher priority traffic class (TC) consumes all the port bandwidth, causing lower priority TCs to run out of credits and their corresponding VOQs to become stuck. This can cause VOQs to stop functioning, potentially leading to traffic loss.

VOQ watchdog actively identifies and resolves issues in VOQs that have not sent packets for over a minute. It detects stuck VOQs in both line cards (LCs) and fabric cards (FCs). The feature is enabled by default and can be disabled using the command **hw-module voq-watchdog feature disable**.

VOQ Watchdog Behavior on Line Cards

By default, the VOQ watchdog feature is enabled on the line cards. The router regularly checks the line cards for packets stuck in VOQs. If the router detects any such packets, it will raise a notification and shut down the line card.

The router displays the following messages when it detects a stuck VOQ.

Syslog Notification - Stuck VOQ; Action: Line card Shutdown

```
LC/0/0/CPU0:Feb 22 09:16:56.090 UTC: npu_drvr[203]: %FABRIC-NPU_DRVR-3-VOQ_HARDWARE_WATCHDOG
 :
[7127] : npu[1]: hardware_watchdog.voq_error: VOQ slc:2 voqnum:19955 isinhbm:1 smscntxtnum:3
 hbmcntxtnum:14
isstuck:1 nochangesec:64 rdptr:1728 wrptr:1735 avblcrdts:-16668 is_fabric:0
```

Stuck VOQ; Action: Line card Shutdown

```
LC/0/0/CPU0:Jan 30 15:10:57.299 UTC: npu_drvr[241]: %PKT_INFRA-FM-2-FAULT_CRITICAL :
ALARM_CRITICAL :VOQ WATCHDOG Alarm :DECLARE :: Shutdown card due to voq watchdog error on
ASIC 1.
```

Line cards automatically shut down if stuck VOQs are detected, making it impossible to identify the root cause of the problem. However, you can prevent the line cards from shutting down by using the **hw-module voq-watchdog cardshut disable** command to disable the shutdown function. This way, the router will send syslog notifications when it detects stuck VOQs without shutting down the line card.

After disabling the shutdown action on the line card, the router displays the following messages when it detects a stuck VOQ.

Syslog Notification - Stuck VOQ; Action: None

```
LC/0/0/CPU0:Feb 22 09:16:56.090 UTC: npu_drvr[203]: %FABRIC-NPU_DRVR-3-VOQ_HARDWARE_WATCHDOG
 :
[7127] : npu[1]: hardware_watchdog.voq_error: VOQ slc:2 voqnum:19955 isinhbm:1 smscntxtnum:3
 hbmcntxtnum:14
isstuck:1 nochangesec:64 rdptr:1728 wrptr:1735 avblcrdts:-16668 is_fabric:0
```

Stuck VOQ; Action: None

```
LC/0/0/CPU0:Feb 22 09:16:56.090 UTC: npu_drvr[203]: %FABRIC-NPU_DRVR-3-VOQ_HARDWARE_WATCHDOG
 :
[7127] : npu[1]: hardware_watchdog.voq_error: VOQ Watchdog Action Handling Skipped Due to
User Configuration
```

VOQ Watchdog Behavior on Fabric Cards

By default, the VOQ watchdog feature is enabled on the fabric cards. The router regularly checks the fabric cards for any packets stuck in VOQs. If such packets are detected, the router raises a syslog notification and hard resets the fabric element (FE) device. After five hard resets, the fabric card undergoes a graceful reload.

The router logs the following syslog notification upon detecting a stuck VOQ on an FC.

```
Syslog: RP/0/RP0/CPU0:Feb 22 09:16:47.721 UTC: npu_drvr[335]:
%FABRIC-NPU_DRVR-3-ASIC_ERROR_ACTION :
[7912] : npu[6]: HARD_RESET needed for hardware_watchdog.voq_error
```

Guidelines and Restrictions for Virtual Output Queue Watchdog

- You can enable or disable the feature on both line and fabric cards. But, you can't enable or disable the feature on just one card, either line or fabric card.
- Disabling the shutdown action for line cards has no effect on fabric cards.
- You can’t disable the shutdown action on fabric cards (FCs).
- The feature is supported only in Cisco 8000 Series Routers with Cisco Silicon One Q200 ASICs.

Configure Virtual Output Queue Watchdog

This section describes how to configure VOQ Watchdog.

Configuration Example

To disable the shutdown action on the line cards:

```
Router# config
Router(config)# hw-module voq-watchdog cardshut disable
Router(config)# commit
```

Running Configuration

```
Router# show running-config
hw-module voq-watchdog cardshut disable
end
```

Verification

The following syslog message is logged on detecting a stuck VOQ on the line card:

```
LC/0/0/CPU0:Feb 22 09:16:56.090 UTC: npu_drvr[203]:
%FABRIC-NPU_DRVR-3-VOQ_HARDWARE_WATCHDOG : [7127] : npu[1]: hardware_watchdog.voq_error:
VOQ slc:2 voqnum:19955 isinhbm:1 smscntxtnum:3 hbmcntxtnum:14 isstuck:1 nochangesec:64
rdptr:1728 wrptr:1735 avblcrdts:-16668 is_fabric:0
```

On disabling the shutdown action of line cards, the following syslog message is logged on detecting a stuck VOQ on a line card:

```
LC/0/0/CPU0:Feb 22 09:16:56.090 UTC: npu_drvr[203]:
%FABRIC-NPU_DRVR-3-VOQ_HARDWARE_WATCHDOG : [7127] : npu[1]:
hardware_watchdog.voq_error: VOQ Watchdog Action Handling Skipped Due to User Configuration
```

The following syslog message is logged on detecting a stuck VOQ on the fabric card:

```
Syslog: RP/0/RP0/CPU0:Feb 22 09:16:47.721 UTC: npu_drvr[335]:
%FABRIC-NPU_DRVR-3-ASIC_ERROR_ACTION :
[7912] : npu[6]: HARD_RESET needed for hardware_watchdog.voq_error
```

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 49
- Random Early Detection and TCP, on page 51

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 49

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 51

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 10: 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*.

CHAPTER 6

Configure Priority Flow Control

Priority Flow Control Overview

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Priority Flow Control on Cisco 8808 and Cisco 8812 Modular Chassis Line Cards	Release 7.5.3	Priority Flow Control is now supported on the following line card in the buffer-internal mode: • 88-LC0-34H14FH The feature is supported in the buffer-internal and buffer-extended modes on: • 88-LC0-36FH Apart from the buffer-external mode, support for this feature now extends to the buffer-internal mode on the following line cards: • 88-LC0-36FH-M • 8800-LC-48H
Shortlink Priority Flow Control	Release 7.3.3	This feature and the hw-module profile priority-flow-control command are supported on 88-LC0-36FH line card.

Feature Name	Release Information	Feature Description
Priority Flow Control Support on Cisco 8800 36x400 GbE QSFP56-DD Line Cards (88-LC0-36FH-M)	Release 7.3.15	This feature and the hw-module profile priority-flow-control command are supported on 88-LC0-36FH-M and 8800-LC-48H line cards. All previous functionalities and benefits of this feature are available on these line cards. However, the buffer-internal mode is not supported. In addition, to use the buffer-extended mode on these line cards, you are required to configure the performance capacity or headroom values. This configuration requirement ensures that you can better provision and balance workloads to achieve lossless behavior, which in turn ensures efficient use of bandwidth and resources.
Priority Flow Control	Release 7.3.1	This feature and the hw-module profile priority-flow-control command are not supported.

Priority-based Flow Control (IEEE 802.1Qbb), which is also referred to as Class-based Flow Control (CBFC) or Per Priority Pause (PPP), is a mechanism that prevents frame loss that is due to congestion. PFC is similar to 802.x Flow Control (pause frames) or link-level flow control (LFC). However, PFC functions on a per class-of-service (CoS) basis.

During congestion, PFC sends a pause frame to indicate the CoS value to pause. A PFC pause frame contains a 2-octet timer value for each CoS that indicates the length of time to pause the traffic. The unit of time for the timer is specified in pause quanta. A quanta is the time required for transmitting 512 bits at the speed of the port. The range is from 0 through 65535 quanta.

PFC asks the peer to stop sending frames of a particular CoS value by sending a pause frame to a well-known multicast address. This pause frame is a one-hop frame and isn't forwarded when received by the peer. When the congestion mitigates, the router stops sending the PFC frames to the upstream node.

You can configure PFC for each line card using the **hw-module profile priority-flow-control** command in one of two modes:

- buffer-internal
- buffer-extended

Note PFC threshold configurations are deprecated in pause command. Use the hw-module profile priority-flow-control command to configure PFC threshold configurations.

Tip You can programmatically retrieve the operational state of the PFC configuration using `Cisco-IOS-XR-ofa-npu-pfc-oper.yang` Cisco IOS XR native data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Related Topics

- Configure Priority Flow Control, on page 61
- Priority Flow Control Watchdog Overview, on page 66

buffer-internal mode

Use this mode if PFC-enabled devices aren't more than 1 km apart.

You can set values for pause-threshold, headroom (both related to PFC), and ECN for the traffic class using the **hw-module profile priority-flow-control** command in this mode. The buffer-internal configuration applies to all ports that the line card hosts, which mean that you can configure a set of these values per line card.

The existing queue limit and ECN configuration in the queueing policy attached to the interface has no impact in this mode.

The effective queue limit for this mode = pause-threshold + headroom (in bytes)

Restrictions and Guidelines

The following restrictions and guidelines apply while configuring the PFC threshold values using the buffer-internal mode.

- The PFC feature isn't supported on fixed chassis systems.
- Ensure that there’s no breakout configured on a chassis that has the PFC configured. Configuring PFC and breakout on the same chassis may lead to unexpected behavior, including traffic loss.
- The feature isn't supported on bundle and non-bundle sub-interface queues.
- The feature is supported on 40GbE, 100 GbE, and 400 GbE interfaces.
- The feature isn't supported in the 4xVOQ queueing mode.
- The feature isn't supported when sharing of VOQ counters is configured.

buffer-extended mode

Use this mode for PFC-enabled devices with long-haul connections.

You can set the value for pause-threshold using the **hw-module profile priority-flow-control** command in this mode. You must, however, configure the queuing policy attached to the interface to set the ECN and queueing limits. The buffer-extended configuration applies to all ports that the line card hosts, which mean that you can configure a set of these values per line card.

Configuration Guidelines

- Important points while configuring the buffer-extended mode on 88-LC0-36FH-M line cards:
  - Apart from pause-threshold, you must also configure values for headroom.
  - The headroom value range is from 4 through 75000.
  - Specify pause-threshold and headroom values in units of kilobytes (KB) or megabytes (MB).
- Important points while configuring the buffer-extended mode on 8800-LC-48H line cards:
  - Configure values only for **pause-threshold**. Don’t configure headroom values.
  - Configure **pause-threshold** in units of milliseconds (ms) or microseconds.
  - Don’t use units of kilobytes (KB) or megabytes (MB) units, even though the CLI displays them as options. Only use units of milliseconds (ms) or microseconds.

(Also see Configure Priority Flow Control, on page 61)

Important Considerations

- If you configure PFC values in the buffer-internal mode, then the ECN value for the line card is derived from the buffer-internal configuration. If you configure PFC values in the buffer-extended mode, then the ECN value is derived from the policy map. (For details on the ECN feature, see Explicit Congestion Notification , on page 53.)
- The buffer-internal and buffer-extended modes can't coexist on the same line card.
- For Cisco 8808 and Cisco 8812 chassis, configure PFC on all line cards in the chassis, regardless of whether you're configuring the buffer-internal or buffer-extended mode. Otherwise, your network may experience traffic loss.
- If you add or remove traffic-class actions on a line card, you must reload the line card.
- When using the buffer-internal mode, you can change values of the following parameters without having to reload the line card. However, if you add a new traffic class and configure these values for the first time on that traffic class, you must reload the line card for the values to come into effect.
  - pause-threshold
  - headroom
  - ECN
- If you add or remove ECN configuration using the **hw-module profile priority-flow-control** command, you must reload the line card for the ECN changes to take effect.
- The PFC threshold value ranges for the buffer-internal mode are as follows.

Threshold	Configured (bytes)
pause (min)	307200
pause (max)	422400
headroom (min)	345600
headroom (max)	537600
ecn (min)	153600
ecn (max)	403200

- For a traffic-class, the ECN value must always be lesser than the configured pause-threshold value.
- The combined configured values for pause-threshold and headroom must not exceed 844800 bytes. Else, the configuration is rejected.
- The pause-threshold value range for buffer-extended mode is from 2 milliseconds (ms) through 25 ms and from 2000 microseconds through 25000 microseconds.

Hardware Support for Priority Flow Control

The table lists the PIDs that support PFC per release and the PFC mode in which the support is available.

Table 12: PFC Hardware Support Matrix

Release	PID	PFC Mode
Release 7.3.15	• 88-LC0-36FH-M • 88-LC0-36FH	buffer-extended
Release 7.0.11	8800-LC-48H	buffer-internal

Configure Priority Flow Control

You can configure PFC to enable the no-drop behavior for the CoS as defined by the active network QoS policy.

Configuration Example

You must accomplish the following to complete the PFC configuration:

1. Enable PFC at the interface level.
2. Configure ingress classification policy.
3. Attach the PFC policy to the interface.
4. Configure PFC threshold values using either the buffer-internal or buffer-extended mode.

```
Router# configure
Router(config)# priority-flow-control mode on
/*Configure ingress classification policy*/
Router(config)# class-map match-any prec7
Router(config-cmap)# match precedence
Router(config)# class-map match-any tc7
/*Ingress policy attach*/
Router(config-if)# service-policy input QOS_marking
/*Egress policy attach*/
Router(config-if)# service-policy output qos_queuing
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)#show controllers npu priority-flow-control location <loc>
```

Running Configuration

```
*Interface Level*
interface HundredGigE0/0/0/0
   priority-flow-control mode on

*Ingress:*
class-map match-any prec7

 match precedence 7

 end-class-map

!

class-map match-any prec6

 match precedence 6

 end-class-map

!

class-map match-any prec5

 match precedence 5

 end-class-map

!

class-map match-any prec4

 match precedence 4

 end-class-map

!

class-map match-any prec3
match precedence 3
end-class-map
!
class-map match-any prec2
match precedence 2
end-class-map
!
class-map match-any prec1
match precedence 1
```

```
end-class-map
!
!
policy-map QOS_MARKING
class prec7
  set traffic-class 7
  set qos-group 7
!
class prec6
  set traffic-class 6
  set qos-group 6
!
class prec5
  set traffic-class 5
  set qos-group 5
!
class prec4
  set traffic-class 4
  set qos-group 4
!
class prec3
  set traffic-class 3
  set qos-group 3
!
class prec2
  set traffic-class 2
  set qos-group 2
!
class prec1
  set traffic-class 1
  set qos-group 1
!
class class-default
  set traffic-class 0
  set qos-group 0
!

*Egress:*
class-map match-any tc7
 match traffic-class 7
 end-class-map
!
class-map match-any tc6
 match traffic-class 6
 end-class-map
!
class-map match-any tc5
 match traffic-class 5
 end-class-map

!

class-map match-any tc4

 match traffic-class 4

 end-class-map

!

class-map match-any tc3

 match traffic-class 3
 end-class-map

!

class-map match-any tc2
match traffic-class 2
end-class-map
!
class-map match-any tc1
match traffic-class 1
end-class-map
!
policy-map QOS_QUEUING
class tc7
  priority level 1
  shape average percent 10
!
class tc6
  bandwidth remaining ratio 1
  queue-limit 100 ms
!
class tc5
  bandwidth remaining ratio 20
  queue-limit 100 ms
!
class tc4
  bandwidth remaining ratio 20
  random-detect ecn
  random-detect 6144 bytes 100 mbytes
!
class tc3
  bandwidth remaining ratio 20
  random-detect ecn
  random-detect 6144 bytes 100 mbytes
!
class tc2
  bandwidth remaining ratio 5
  queue-limit 100 ms
!
class tc1
  bandwidth remaining ratio 5
  queue-limit 100 ms
!
class class-default
  bandwidth remaining ratio 20
  queue-limit 100 ms
!
[buffer-extended]

hw-module profile priority-flow-control location 0/0/CPU0
 buffer-extended traffic-class 3 pause-threshold 10 ms
 buffer-extended traffic-class 4 pause-threshold 10 ms
!

[buffer-internal]

hw-module profile priority-flow-control location 0/1/CPU0
 buffer-internal traffic-class 3 pause-threshold 403200 bytes headroom 441600 bytes ecn
224640 bytes
 buffer-internal traffic-class 4 pause-threshold 403200 bytes headroom 441600 bytes ecn
224640 bytes
```

Verification

```
Router#sh controllers hundredGigE0/0/0/22 priority-flow-control
Priority flow control information for interface HundredGigE0/0/0/22:
Priority Flow Control:
Total Rx PFC Frames : 0
Total Tx PFC Frames : 313866
Rx Data Frames Dropped: 0
CoS Status Rx Frames
--- ------ ----------
0   on     0
1   on     0
2   on     0
3   on     0
4   on     0
5   on     0
6   on     0
7   on     0

/*[buffer-internal]*/
Router#show controllers hundredGigE 0/9/0/24 priority-flow-control

Priority flow control information for interface HundredGigE0/9/0/24:

Priority Flow Control:
Total Rx PFC Frames : 0
Total Tx PFC Frames : 313866
Rx Data Frames Dropped: 0
CoS Status Rx Frames
--- ------ ----------
0   on     0
1   on     0
2   on     0
3   on     0
4   on     0
5   on     0
6   on     0
7   on     0
…

/*[buffer-internal, tc3 & tc4 configured. TC4 doesn’t have ECN]*/

Router#show controllers npu priority-flow-control location <loc>
Location Id:                        0/1/CPU0
PFC:                                Enabled
PFC-Mode:                           buffer-internal
TC   Pause           Headroom        ECN
-------------------------------------------------------
3    86800 bytes     120000 bytes    76800 bytes
4    86800 bytes     120000 bytes    Not-configured

/*[buffer-extended PFC, tc3 & tc4 configured]*/

Router#show controllers npu priority-flow-control location <loc>
Location Id:                        0/1/CPU0
PFC:                                Enabled
PFC-Mode:                           buffer-extended
TC   Pause
-----------
3    5000 us
4    10000 us

/*[No PFC]*/
```

```
Router#show controllers npu priority-flow-control location <loc>
Location Id:                        0/1/CPU0
PFC:                                Disabled
```

Related Topics

- Priority Flow Control Overview, on page 57

Related Commands hw-module profile priority-flow-control location

Priority Flow Control Watchdog Overview

PFC Watchdog is a mechanism to identify any PFC storms (queue-stuck condition) in the network. It also prevents the PFC from propagating on the network and running in a loop. You can configure a PFC watchdog interval to detect whether packets in a no-drop queue are drained within a specified time period. When the time period is exceeded, all outgoing packets are dropped on interfaces that match the PFC queue that is not being drained.

This requires monitoring PFC receiving on each port and detecting ports seeing an unusual number of sustained pause frames. Once detected, the watchdog module can enforce several actions on such ports, which include generating a syslog message for network management systems, shutting down the queue, and autorestoring the queue (after the PFC storm stops).

Here's how the PFC Watchdog works:

1. The Watchdog module monitors the PFC-enabled queues to determine the reception of an unusual amount of PFC pause frames in a given interval (Watchdog interval.)
2. Your hardware notifies the Watchdog module when too many PFC frames are received and traffic on the corresponding queues is halted for a time interval.
3. On receiving such notifications, the Watchdog module starts the shutdown timer and moves the queue state to wait-to-shutdown state.
4. At regular intervals during the shutdown interval, the queue is checked for PFC frames and if the traffic in the queue is stuck. If the traffic isn't stuck because the queue didn't receive any PFC frames, the queue moves back to the monitored state.
5. If the traffic is stuck for a longer time and the shutdown-timer expires, the queue switches to a drop state and the PFC Watchdog begins to drop all packets.
6. At regular intervals, the Watchdog checks the queue for PFC frames and whether the traffic in the queue is still stuck. If traffic is stuck in the queue as PFC packets keep arriving, the queue remains in the drop or shutdown state.
7. When the traffic's no longer stuck, the autorestore timer starts. At regular intervals, the module checks if the queue is stuck because of PFC frames.
8. If the queue receives PFC frames during the last autorestore interval, the auto-restore timer is reset upon expiry.
9. If the queue receives no PFC frames during the last autorestore interval, the Watchdog module restores the queue, and traffic resumes.

Related Topics

- Priority Flow Control Overview, on page 57

Configure a Priority Flow Control Watchdog Interval

You can configure PFC Watchdog parameters (Watchdog interval, shutdown multiplier, auto-restore multiplier) at the global or interface levels. Note that:

- When global Watchdog mode is disabled or off, Watchdog is disabled on all interfaces. This condition is regardless of the interface level Watchdog mode settings.
- When global Watchdog mode is enabled or on, the interface level Watchdog mode configuration settings override the global Watchdog mode values.
- When you configure interface level Watchdog attributes such as interval, shutdown multiplier, and auto-restore multiplier, they override the global Watchdog attributes.

Note Configuring the PFC mode and its policies is a prerequisite for PFC Watchdog.

Configuration Example

You can configure the Watchdog at the global or at the interface level.

Note Watchdog is enabled by default, with system default values of:

Watchdog interval = 100 ms

Shutdown multiplier = 1

Auto-restore multiplier = 10

```
P/0/RP0/CPU0:ios#show controllers hundredGigE 0/2/0/0 priority-flow-control

Priority flow control information for interface HundredGigE0/2/0/0:

Priority flow control watchdog configuration:
(D) : Default value
U : Unconfigured
--------------------------------------------------------------------------------
Configuration Item          Global   Interface   Effective
--------------------------------------------------------------------------------
PFC watchdog state :          U        U         Enabled(D)
Poll interval :               U        U         100(D)
Shutdown multiplier :         U        U         1(D)
Auto-restore multiplier :     U        U         10(D)
RP/0/RP0/CPU0:ios#config
RP/0/RP0/CPU0:ios(config)#priority-flow-control watchdog mode off
RP/0/RP0/CPU0:ios(config)#commit

RP/0/RP0/CPU0:ios(config)#do show controllers hundredGigE 0/2/0/0 priority-flo$
```

```
Priority flow control information for interface HundredGigE0/2/0/0:


Priority flow control watchdog configuration:
(D) : Default value
U : Unconfigured
--------------------------------------------------------------------------------
Configuration Item          Global  Interface  Effective
--------------------------------------------------------------------------------
PFC watchdog state :        Disabled  U        Disabled
Poll interval :              U        U        100(D)
Shutdown multiplier :        U        U        1(D)
Auto-restore multiplier :    U        U        10(D)

RP/0/RP0/CPU0:ios(config)#interface hundredGigE 0/2/0/0 priority-flow-control $
RP/0/RP0/CPU0:ios(config)#commit

RP/0/RP0/CPU0:ios(config)#do show controllers hundredGigE 0/2/0/0 priority-flo$

Priority flow control information for interface HundredGigE0/2/0/0:


Priority flow control watchdog configuration:
(D) : Default value
U : Unconfigured
--------------------------------------------------------------------------------
Configuration Item             Global   Interface    Effective
--------------------------------------------------------------------------------
PFC watchdog state :           Disabled  Enabled      Disabled
Poll interval :                  U         U          100(D)
Shutdown multiplier :            U         U          1(D)
Auto-restore multiplier :        U         U          10(D)
RP/0/RP0/CPU0:ios(config)#interface hundredGigE 0/2/0/1 priority-flow-control $
RP/0/RP0/CPU0:ios(config)#commit

RP/0/RP0/CPU0:ios(config)#do show controllers hundredGigE 0/2/0/1 priority-flo$

Priority flow control information for interface HundredGigE0/2/0/1:


Priority flow control watchdog configuration:
(D) : Default value
U: Unconfigured
--------------------------------------------------------------------------------
Configuration Item               Global   Interface   Effective
--------------------------------------------------------------------------------
PFC watchdog state :             Enabled  Disabled     Disabled
Poll interval :                    U        U          100(D)
Shutdown multiplier :              U        U          1(D)
Auto-restore multiplier :          U        U          10(D)
```

Related Topics

- Priority Flow Control Watchdog Overview, on page 66

CHAPTER 7

Congestion Management

Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which a traffic flow (or packets) is sent out an interface based on priorities assigned to packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission.

The types of traffic regulation mechanisms supported are:

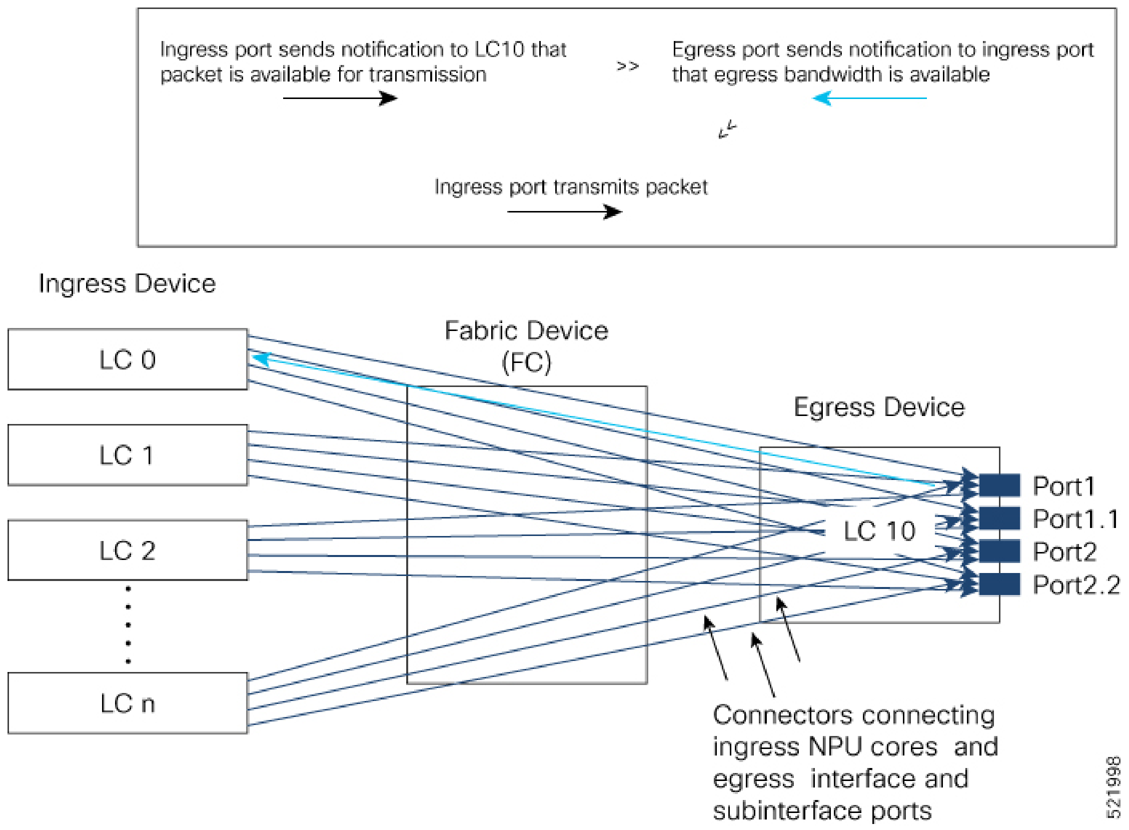
- Low Latency Queueing with Strict Priority Queueing, on page 71
- Traffic Shaping, on page 73
- Traffic Policing, on page 75

Ingress Traffic Management Model

The ingress traffic management model relies on packet queueing on the egress interface using Virtual Output Queueing (VOQ) on the ingress. In this model, buffering takes place at ingress. Here’s how the VOQ process works.

Your routers support up to eight output queues per main interface or physical port. For every egress output queue, the VOQ model earmarks buffer space on every ingress pipeline. This buffer space is in the form of dedicated VOQs. These queues are called virtual because the queues physically exist on the ingress interface only when the line card actually has packets enqueued to it. To support the modular model of packet distribution, each network processing unit (NPU) core at the ingress needs connectors to every egress main interface and subinterface. The ingress traffic management model thus requires a mesh of connectors to connect the ingress NPU cores to the egress interfaces, as shown in **The Ingress Traffic Management Model**.

Figure 4: The Ingress Traffic Management Model


Ingress port sends notification to LC10 that packet is available for transmission
Egress port sends notification to ingress port that egress bandwidth is available
Ingress port transmits packet
Ingress Device
LC 0
LC 1
LC 2
LC n
Fabric Device (FC)
Egress Device
LC 10
Port1
Port1.1
Port2
Port2.2
Connectors connecting ingress NPU cores and egress interface and subinterface ports
521998

In the figure, every ingress interface (LC 0 through LC n) port has eight VOQs for the single egress line card LC 10.

Here's how packet transmission takes place:

1. When a packet arrives at ingress port (say on LC 0), the forwarding lookup on ingress line card points to the egress interface. Based on the egress interface (say it is on LC10), the packet is enqueued to the VOQ of LC 10. The egress interface is always mapped to a physical port.
2. Once egress bandwidth is available, the LC 10 ports ready to receive the packets (based on the packet marking and distribution model) send grants to the ingress ports via the connectors. (The figure shows a separate line for the grant for the sake of visual representation. In reality, the same connector is used for requests, grants, and transmission between an NPU core at the ingress and the egress port on LC 10.)
3. The ingress ports respond to this permission by transmitting the packets via FC to the LC 10 ports. (The time it takes for the ingress ports to request for egress port access, the egress port to grant access, and the packet to travel across FC is the round-trip time.)

The VOQ model thus operates on the principle of storing excess packets in buffers at ingress until bandwidth becomes available. Based on the congestion that builds up and the configured threshold values, packets begin to drop at the ingress itself, instead of having to travel all the way to the egress interface and then getting dropped.

Hardware Limitation:

In a scale scenario where 1000+ VoQs (created using egress QoS policies) store packets due to active traffic flows and may consume all the available on-chip buffer (OCB), unexpected traffic drops will be seen even though the traffic rate at the VoQ level is less than that of the VoQ shaper.

Low Latency Queueing with Strict Priority Queueing

The Low-Latency Queueing (LLQ) feature brings strict priority queuing (PQ) to the CBWFQ scheduling mechanism. Priority Queueing (PQ) in strict priority mode ensures that one type of traffic is sent, possibly at the expense of all others. For PQ, a low-priority queue can be detrimentally affected, and, in the worst case, never allowed to send its packets if a limited amount of bandwidth is available or the transmission rate of critical traffic is high. Strict PQ allows delay-sensitive data, such as voice, to be de-queued and sent before packets in other queues are de-queued.

Configure Low Latency Queueing with Strict Priority Queueing

Configuring low latency queueing (LLQ) with strict priority queuing (PQ) allows delay-sensitive data such as voice to be de-queued and sent before the packets in other queues are de-queued.

Guidelines

- Only priority level 1 to 7 is supported, with 1 being the highest priority and 7 being the lowest. However, the default CoSQ 0 has the lowest priority among all.
- Egress policing is not supported. Hence, in the case of strict priority queuing, there are chances that the other queues do not get serviced.
- You can configure **shape average** and **queue-limit** commands along with **priority**.
- You can configure an optional shaper to cap the maximum traffic rate. This is to ensure the lower priority traffic classes are not starved of bandwidth.

Configuration Example

You have to accomplish the following to complete the LLQ with strict priority queuing:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Specifying priority to the traffic class
4. (Optional) Shaping the traffic to a specific bit rate
5. Attaching the policy-map to an output interface

```
Router# configure
Router(config)# policy-map test-priority-1
Router(config-pmap)# class qos1
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-priority-1
Router(config-if)# no shutdown
Router(config-if)# commit
```

Running Configuration

```
policy-map strict-priority
class tc7
  priority level 1
  queue-limit 75 mbytes
!
 class tc6
  priority level 2
  queue-limit 75 mbytes
!
 class tc5
  priority level 3
  queue-limit 75 mbytes
 !
 class tc4
  priority level 4
  queue-limit 75 mbytes
!
 class tc3
  priority level 5
  queue-limit 75 mbytes
!
 class tc2
  priority level 6
  queue-limit 75 mbytes
 !
 class tc1
  priority level 7
  queue-limit 75 mbytes
 !
 class class-default
  queue-limit 75 mbytes
 !
 end-policy-map
!



class-map match-any tc1
match traffic-class 1
 end-class-map
!
class-map match-any tc2
match traffic-class 2
 end-class-map
!
class-map match-any tc3
match traffic-class 3
 end-class-map
!
class-map match-any tc4
match traffic-class 4
 end-class-map
!
class-map match-any tc5
match traffic-class 5
 end-class-map
!
class-map match-any tc6
match traffic-class 6
```

```
 end-class-map
!
class-map match-any tc7
match traffic-class 7
 end-class-map
!

interface HundredGigE0/6/0/18
 service-policy input 100g-s1-1
 service-policy output test-priority-1
!
```

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:       3
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                       =   125304832 bytes / 10 ms (default)
WRED not configured for this class

Level1 Class (HP6)                       =   qos-2
Egressq Queue ID                         =   11178 (HP6 queue)
TailDrop Threshold                       =   125304832 bytes / 10 ms (default)
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

- Congestion Management Overview, on page 69
- Traffic Shaping, on page 73

Traffic Shaping

Traffic shaping allows you to control the traffic flow exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it. Traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

Note Traffic shaping is supported only in egress direction.

Configure Traffic Shaping

The traffic shaping performed on outgoing interfaces is done at the Layer 1 level and includes the Layer 1 header in the rate calculation.

Guidelines

- Only egress traffic shaping is supported.
- It is mandatory to configure all the eight qos-group classes (including class-default) for the egress policies.
- You can configure **shape average** command along with **priority** command.
- It is recommended that you configure all the eight <traffic-class classes> (including **class-default**) for the egress policies. A limited number of < traffic-class class> combinations are supported, but unicast and multicast traffic-class may not be handled consistently. Hence, such configurations are recommended, when the port egress has only unicast traffic.

Configuration Example

You have to accomplish the following to complete the traffic shaping configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Shaping the traffic to a specific bit rate
4. Attaching the policy-map to an output interface

```
Router# configure
Router(config)# policy-map egress_policy1
Router(config-pmap)# class c5
Router(config-pmap-c)# shape average percent 40
Router(config-pmap-c# exit
Router(config-pmap)# exit
Router(config)# interface HundredGigE 0/1/0/0
Router(config-if)# service-policy output egress_policy1
Router(config-if)# commit
```

Running Configuration

```
policy-map egress_policy1
 class c5
  shape average percent 40
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE0/6/0/18
```

```
 service-policy input 100g-s1-1
 service-policy output egress_policy1
!
```

Verification

```
Router#  show qos interface 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                             =   c5
Egressq Queue ID                         =   11177 (LP queue)
Queue Max. BW.                           =   40329846 kbps (40 %)
Queue Min. BW.                           =   0 kbps (default)
Inverse Weight / Weight                  =   1 (BWR not configured)
Guaranteed service rate                  =   40000000 kbps
TailDrop Threshold                       =   50069504 bytes / 10 ms (default)
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)
Guaranteed service rate                  =   50000000 kbps
TailDrop Threshold                       =   62652416 bytes / 10 ms (default)
WRED not configured for this class
```

Related Topics

- Congestion Management Overview, on page 69

Traffic Policing

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface and to partition a network into multiple priority levels or class of service (CoS).Traffic policing manages the maximum rate of traffic through a token bucket algorithm. The token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on an interface at a given moment in time. The token bucket algorithm is affected by all traffic entering or leaving the interface (depending on where the traffic policy with traffic policing is configured) and is useful in managing network bandwidth in cases where several large packets are sent in the same traffic stream. By default, the configured bandwidth value takes into account the Layer 2 encapsulation that is applied to traffic leaving the interface.

The router supports the following traffic policing mode:

- Single-Rate Two-Color (SR2C) in color-blind mode. See Single-Rate Policer, on page 76.

Restrictions

- 1R3C policers are not supported.
- Up to eight policers are supported per ingress policy.
- Policers are allocated in multiples of 2, so any request for allocating odd number of policers is internally rounded up by a factor of one.
- Only one conditional marking action is supported. You can set discard class to 0/1, that is used to set either virtual output queueing (VOQ) limits or Random Early Detection (RED) profiles.
- If you configure discard-class explicitly at the class level and not under policer, then the explicit mark action is applied to all the transmitted policer traffic.
- Egress policing is not supported.

Committed Bursts and Excess Bursts

Unlike a traffic shaper, a traffic policer does not buffer excess packets and transmit them later. Instead, the policer executes a "send or do not send" policy without buffering. Policing uses normal or committed burst (bc) values and excess burst values (be) to ensure that the router reaches the configured committed information rate (CIR). Policing decides if a packet conforms or exceeds the CIR based on the burst values you configure. Burst parameters are based on a generic buffering rule for routers, which recommends that you configure buffering to be equal to the round-trip time bit-rate to accommodate the outstanding TCP windows of all connections in times of congestion. During periods of congestion, proper configuration of the excess burst parameter enables the policer to drop packets less aggressively.

For more details, see Committed Bursts, on page 81 and Excess Bursts, on page 82.

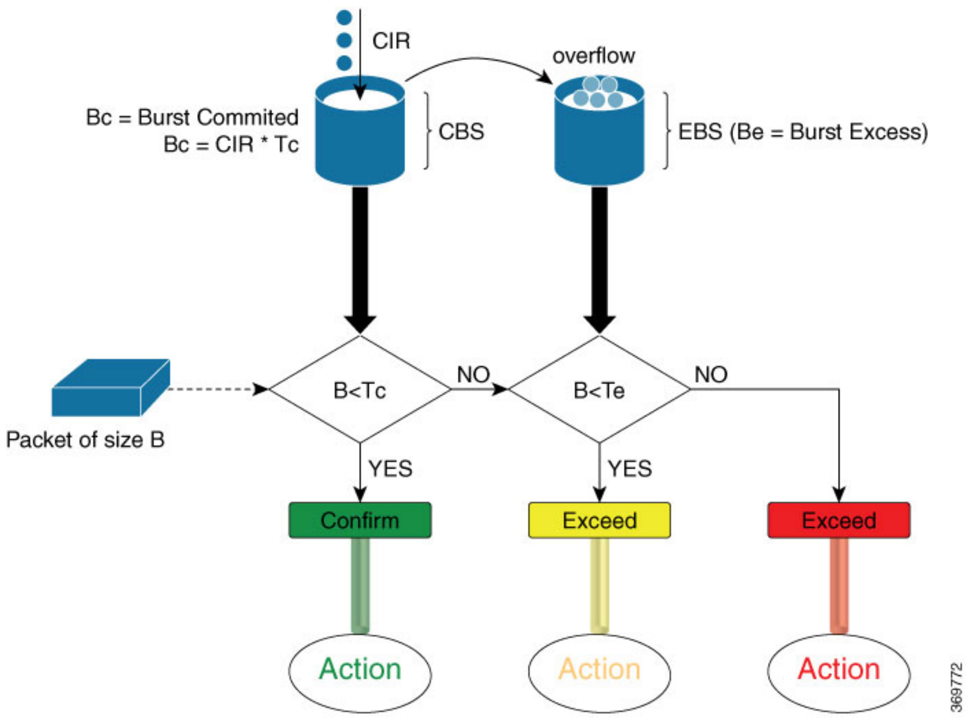
Single-Rate Policer

This section explains the concept of the single-rate two-color policer.

Single-Rate Two-Color Policer

A single-rate two-color (1R2C) policer provides one token bucket with two actions for each packet: a conform action and an exceed action.

Figure 5: Workflow of Single-Rate Two-Color Policey



Based on the committed information rate (CIR) value, the token bucket is updated at every refresh time interval. The Tc token bucket can contain up to the Bc value, which can be a certain number of bytes or a period of time. If a packet of size B is greater than the Tc token bucket, then the packet exceeds the CIR value and a default action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different default action is performed.

Configure Traffic Policing (Single-Rate Two-Color)

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. The default conform action for single-rate two color policer is to transmit the packet and the default exceed action is to drop the packet. Users cannot modify these default actions.

Configuration Example

You have to accomplish the following to complete the traffic policing configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. (Optional) Specifying the marking action
4. Specifying the policy rate for the traffic
5. Attaching the policy-map to an input interface

```
Router# configure
Router(config)# policy-map test-police-1R2C
Router(config-pmap)# class dscp1
Router(config-pmap-c)# police rate 10 gbps
Router(config-pmap-c-police)# exit
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface hundredGigE 0/0/0/18
Router(config-if)# service-policy input test-police-1R2C
Router(config-if)# commit
```

Running Configuration

```
class-map match-any dscp1
 match dscp ipv4 1
 end-class-map
!
!
policy-map test-police-1R2C
 class dscp1
  police rate 10 gbps
  !
 !
 class class-default
  !
 !
 end-policy-map
!
!
interface HundredGigE0/0/0/8
service-policy input test-police-1R2C
!
```

Verification

```
Router# show qos interface hundredGigE 0/0/0/8 input
NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/8 ifh 0xf0001e8  -- input policy
NPU Id:                      0
Total number of classes:     2
Interface Bandwidth:         100000000 kbps
Policy Name:                 test-police-1R2C
Accounting Type:             Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                          =   dscp1
Policer committed rate                =   10000000 kbps (10 gbits/sec)
Policer conform burst                 =   1024000 bytes (default)
Policer conform action                =   Just TX
Policer exceed action                 =   DROP PKT

Level1 Class                          =   class-default
Policer not configured for this class
```

Related Topics

- Traffic Policing, on page 75

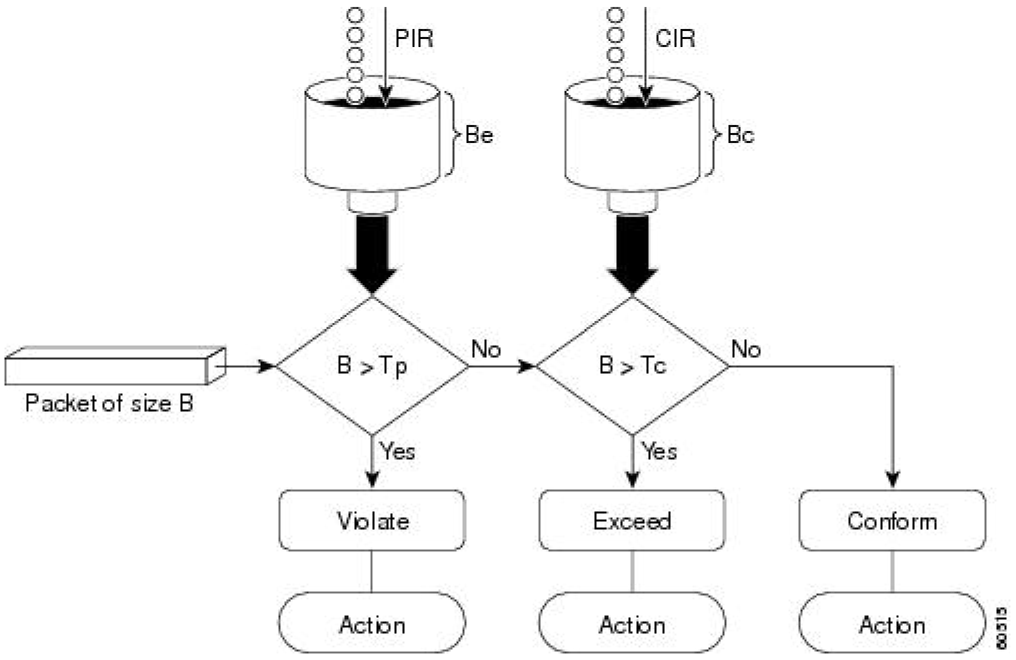
Two-Rate Policer

The two-rate policer manages the maximum rate of traffic by using two token buckets: the committed token bucket and the peak token bucket. The dual-token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on a queue at a given moment. In this way, the two-rate policer can meter traffic at two independent rates: the committed information rate (CIR) and the peak information rate (PIR).

The dual-token bucket algorithm provides users with three actions for each packet—a conform action, an exceed action, and an optional violate action. Traffic entering a queue with the two-rate policer configured is placed into one of these categories. The actions are pre-determined for each category. The default conform and exceed actions are to transmit the packet, and the default violate action is to drop the packet.

This figure shows how the two-rate policer marks a packet and assigns a corresponding action to the packet.

Figure 6: Marking Packets and Assigning Actions—Two-Rate Policer



Also, see Two-Rate Policer Details, on page 82.

The router supports Two-Rate Three-Color (2R3C) policer.

Configure Traffic Policing (Two-Rate Three-Color)

The default conform and exceed actions for two-rate three-color (2R3C) policer are to transmit the packet and the default violate action is to drop the packet. Users cannot modify these default actions.

Configuration Example

You have to accomplish the following to complete the two-rate three-color traffic policing configuration:

1. Creating or modifying a policy-map that can be attached to one or more interfaces
2. Specifying the traffic class whose policy has to be created or changed
3. Specifying the packet marking

4. Configuring two rate traffic policing
5. Attaching the policy-map to an input interface

```
Router# configure
Router(config)# policy-map test-police-2R3C
Router(config-pmap)# class dscp1
Router(config-pmap-c)# police rate 10 gbps peak-rate 20 gbps
Router(config-pmap-c-police)# exit
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# iinterface hundredGigE 0/0/0/8
Router(config-if)# service-policy input test-police-2R3C
Router(config-if)# commit
```

Running Configuration

```
class-map match-any dscp1
 match dscp ipv4 1
 end-class-map
!
!
policy-map test-police-2R3C
 class dscp1
  police rate 10 gbps peak-rate 20 gbps
  !
 !
 class class-default
  !
 !
 end-policy-map
!
!
interface HundredGigE0/0/0/8
service-policy input test-police-2R3C
!
```

Verification

```
Router# show qos interface hundredGigE 0/0/0/8 input

NOTE:- Configured values are displayed within parentheses
Interface HundredGigE0/0/0/8 ifh 0xf0001e8  -- input policy
NPU Id:                        0
Total number of classes:       2
Interface Bandwidth:           100000000 kbps
Policy Name:                   test-police-2R3C
Accounting Type:               Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class                             =   dscp1
Policer committed rate                   =   10000000 kbps (10 gbits/sec)
Policer peak rate                        =   20000000 kbps (20 gbits/sec)
Policer conform burst                    =   1024000 bytes (default)
Policer exceed burst                     =   2048000 bytes (default)
Policer conform action                   =   Just TX
Policer exceed action                    =   DROP PKT
Policer violate action                   =   DROP PKT
```

```
Level1 Class                             =   class-default
Policer not configured for this class


Router# policy-map interface hundredGigE 0/0/0/8 input

HundredGigE0/0/0/8 input: test-police-2R3C

Class dscp1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          289228439/289228439000         27734775
    Transmitted         :           56422213/56422213000          5410359
    Total Dropped       :          232806226/232806226000         22324416
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :           56422213/56422213000          5410359
    Policed(exceed)     :           56422215/56422215000          5410358
    Policed(violate)    :          176384011/176384011000         16914058
    Policed and dropped :          232806226/232806226000
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           61136620/61136620000          0
    Transmitted         :           61136620/61136620000          0
    Total Dropped       :                  0/0                    0
Policy Bag Stats time: 1570155764000  [Local Time: 10/04/19 02:22:44.000]
```

Related Topics

- Two-Rate Policer, on page 79

Committed Bursts

The committed burst (bc) parameter of the police command implements the first, conforming (green) token bucket that the router uses to meter traffic. The bc parameter sets the size of this token bucket. Initially, the token bucket is full and the token count is equal to the committed burst size (CBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the conforming token bucket to send packets:

- If sufficient tokens are in the conforming token bucket when a packet arrives, the meter marks the packet green and decrements the conforming token count by the number of bytes of the packet.
- If there are insufficient tokens available in the conforming token bucket, the meter allows the traffic flow to borrow the tokens needed to send the packet. The meter checks the exceeding token bucket for the number of bytes of the packet. If the exceeding token bucket has a sufficient number of tokens available, the meter marks the packet.

  Green and decrements the conforming token count down to the minimum value of 0.

  Yellow, borrows the remaining tokens needed from the exceeding token bucket, and decrements the exceeding token count by the number of tokens borrowed down to the minimum value of 0.
- If an insufficient number of tokens is available, the meter marks the packet red and does not decrement either of the conforming or exceeding token counts.

Note When the meter marks a packet with a specific color, there must be a sufficient number of tokens of that color to accommodate the entire packet. Therefore, the volume of green packets is never smaller than the committed information rate (CIR) and committed burst size (CBS). Tokens of a given color are always used on packets of that color.

Excess Bursts

The excess burst (be) parameter of the police command implements the second, exceeding (yellow) token bucket that the router uses to meter traffic. The exceeding token bucket is initially full and the token count is equal to the excess burst size (EBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the exceeding token bucket to send packets:

- When the first token bucket (the conforming bucket) meets the committed burst size (CBS), the meter allows the traffic flow to borrow the tokens needed from the exceeding token bucket. The meter marks the packet yellow and then decrements the exceeding token bucket by the number of bytes of the packet.
- If the exceeding token bucket does not have the required tokens to borrow, the meter marks the packet red and does not decrement the conforming or the exceeding token bucket. Instead, the meter performs the exceed-action configured in the police command (for example, the policer drops the packets).

Important Points

- User configurable burst values— committed burst size (CBS) and excess burst size (EBS)—are not supported. Instead, they are derived using user configured rates—committed information rates (CIR) and peak information rates (PIR).
- The router supports two burst values: low (10 kilobytes) and high (1 megabyte). For a lower range of configured values (1.6 MBps to less than 1 GBps), the burst value is 10 KB. For a higher range of configured values (1 GBps to 400 GBps), the burst value is 1 MB.

Two-Rate Policer Details

The committed token bucket can hold bytes up to the size of the committed burst (bc) before overflowing. This token bucket holds the tokens that determine whether a packet conforms to or exceeds the CIR as the following describes:

- A traffic stream is conforming when the average number of bytes over time does not cause the committed token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream green.
- A traffic stream is exceeding when it causes the committed token bucket to overflow into the peak token bucket. When this occurs, the token bucket algorithm marks the traffic stream yellow. The peak token bucket is filled as long as the traffic exceeds the police rate.

The peak token bucket can hold bytes up to the size of the peak burst (be) before overflowing. This token bucket holds the tokens that determine whether a packet violates the PIR. A traffic stream is violating when it causes the peak token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream red.

For example, if a data stream with a rate of 250 kbps arrives at the two-rate policer, and the CIR is 100 kbps and the PIR is 200 kbps, the policer marks the packet in the following way:

- 100 kbps conforms to the rate
- 100 kbps exceeds the rate
- 50 kbps violates the rate

The router updates the tokens for both the committed and peak token buckets in the following way:

- The router updates the committed token bucket at the CIR value each time a packet arrives at the interface. The committed token bucket can contain up to the committed burst (bc) value.
- The router updates the peak token bucket at the PIR value each time a packet arrives at the interface. The peak token bucket can contain up to the peak burst (be) value.
- When an arriving packet conforms to the CIR, the router takes the conform action on the packet and decrements both the committed and peak token buckets by the number of bytes of the packet.
- When an arriving packet exceeds the CIR, the router takes the exceed action on the packet, decrements the committed token bucket by the number of bytes of the packet, and decrements the peak token bucket by the number of overflow bytes of the packet.
- When an arriving packet exceeds the PIR, the router takes the violate action on the packet, but does not decrement the peak token bucket.

See Two-Rate Policer, on page 79.

References for Modular QoS Management

Read this section for more information on committed bursts, excess bursts, and two-rate policer.

CHAPTER 8

Configure Modular QoS on Link Bundles

QoS on Link Bundles

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The router supports Ethernet interfaces and VLAN interfaces (bundle sub-interfaces) bundles. All QoS features currently supported on physical interfaces, are also supported on all link bundle interfaces. Applying QoS on bundle members is not supported.

Restrictions for Link Bundles

- Only Ethernet link bundling is supported.
- Only relative (percentage-based) shaper values can be configured. You cannot configure absolute values
- A bundle interface can only contain physical interface.
- All links within a single bundle must be configured either to run 802.3ad (LACP) or EtherChannel (non-LACP). Mixed links within a single bundle are not supported.
- MAC accounting is not supported on Ethernet link bundles.
- Maximum number of links supported in each link bundle is 64.
- The maximum number of link bundles supported is 128.

Load Balancing

Load balancing function is a forwarding mechanism to distribute traffic over multiple links based on Layer 3 routing information in the router. Per-destination load balancing isonly supported on the router, where the router is allowed to distribute packets over one of the links in the bundle. When the per-destination load balancing is enabled, all packets for a certain source-destination pair goes through the same link, though there are multiple links available. In other words, per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

Layer 3 Load Balancing on Link Bundles

Layer 3 load balancing for link bundles is done on Ethernet Flow Points (EFPs) and is based on the IPv4 source and destination addresses in the packet. When Layer 3 service-specific load balancing is configured, all egress bundles are load balanced based on the IPv4 source and destination addresses. When packets do not have IPv4 addresses, default load-balancing (based on the MAC SA/DA fields in the packet header) is used.

Configure QoS on Link Bundles

QoS is configured on link bundles in the same way that it's configured on individual interfaces.

Guidelines

- When you apply a QoS policy on a bundle (ingress or egress direction), the queuing policy is applied at each member interface. The reference bandwidth that is used to calculate shaper or bandwidth values is applied as per the physical member interface bandwidth.
- If a QoS policy is not applied to a bundle interface, both the ingress and egress traffic use the default queue of the per link member port.
- The shape rate specified in the bundle policy-map is not an aggregate for all bundle members. The shape rate applied to the bundle depends on the load balancing of the links. For example, if a policy map with a shape rate of 10 Mbps is applied to a bundle with two member links, and if the traffic is always load-balanced to the same member link, then an overall rate of 10 Mbps applies to the bundle. However, if the traffic is load-balanced evenly between the two links, the overall shape rate for the bundle becomes 20 Mbps.
- If a member is deleted from a bundle, the total bundle statistics changes because the statistics that belongs to the detached link is lost.
- Add bundle members before you apply the QoS policy to the bundle. This action ensures that the QoS policy applies across the bundle and on the new member. If a QoS policy exists and you add a new member later, the policy may not apply on the bundle if the member interface has an incompatible configuration. This scenario may result in errors.
- **Queue-limit error:**
  - If you experience a queue-limit error, remove the existing QoS policy from the link bundle interfaces. However, this removal may take up to ten minutes to come into effect, depending on the time lapsed since the error displayed. To see the time lapsed, check the **retries performed** information in the subsequent syslog.
  - When you encounter an invalid queue-limit error, the syslog lists only one interface, though the error may occur across multiple interfaces. To know the number of affected interfaces, see **operations** in the syslog.

  For example:

```
RP/0/RP0/CPU0:Aug 17 09:56:37.417 PDT: BM-DISTRIB[1156]:
%L2-BM-4-ERR_OP_RETRY_THRESHOLD :
Exceeded threshold for the number of retries for 4 operations on members.
First (and error) FortyGigE0/0/0/34 ('DPA_QOSEA' detected the 'warning'
condition 'Invalid Queue Limit.
Value should be in range 614400-390070272 bytes or equivalent value in usec or
msec based on the service rate of the queue'),
retries performed for 10 hrs 16 mins

RP/0/RP0/CPU0:Aug 17 10:06:40.388 PDT: BM-DISTRIB[1156]:
%L2-BM-4-ERR_OP_RETRY_THRESHOLD :
Exceeded threshold for the number of retries for 4 operations on members.
First (and error) FortyGigE0/0/0/34 ('DPA_QOSEA' detected the 'warning'
condition 'Invalid Queue Limit.
Value should be in range 614400-390070272 bytes or equivalent value in
usec or msec based on the service rate of the queue'),
retries performed for 10 hrs 26 mins
```

- The QoS policy applied on bundle is inherited to all its member links and the reference bandwidth used to calculate shaper/bandwidth is applied as per the physical member interface bandwidth, and not the bundle as a whole.
- Classification resources (TCAM/MAP) and counters are allocated per bundle interface, not per member.
- The policing policy can be applied on bundle main interface or sub interface, but not on both.
- Policer committed information rates (CIR) and peak information rates (PIR) can be configured only in percent units.
- Policers and policer counters are allocated per bundle interface but not per member.
- The queuing policy can be applied on bundle main or bundle sub interface, but not on both.
- Queuing resources (VoQs) and counters are allocated per bundle member interface.
- The classification and marking policies can be applied on the bundle main or the bundle sub interface, but not on both.

Configuration Example

You have to accomplish the following to complete the QoS configuration on link bundles:

1. Creating a class-map
2. Creating a policy-map and specifying the respective class-map
3. Specifying the action type for the traffic

   Refer Attach a Traffic Policy to an Interface, on page 21 for details on step 1, 2 and 3.
4. Creating a link bundle
5. Applying traffic policy to the link bundle

```
/* Configure Ether-Bundle and apply traffic policy */
Router(config)# interface Bundle-Ether 12000
Router(config-if)# mtu 9100
Router(config-if)# service-policy input ingress
Router(config-if)# service-policy output egress
Router(config-if)# ipv4 address 100.12.0.0 255.255.255.254
Router(config-if)# bundle maximum-active links 64
Router(config-if)# commit
```

Running Configuration

This example shows how a queuing policy is applied on an Ethernet link bundle. The policy is applied to all interfaces that are members of the Ethernet link bundle.

```
/* Policy-map */

policy-map ingress
 class inet4-classifier-af1
  set qos-group 1
 !
 class inet4-classifier-af2
  set qos-group 2
 !
 class inet4-classifier-af3
  set qos-group 3
 !
 class inet4-classifier-af4
  set qos-group 4
 !
 class inet4-classifier-be1
  set qos-group 5
 !
 class inet4-classifier-nc1
  set qos-group 6
 !
 class class-default
 !
 end-policy-map
!

/* Ether Bundle */
interface Bundle-Ether12000
 mtu 9100
 service-policy input ingress
 service-policy output egress
 ipv4 address 100.12.0.0 255.255.255.254
 load-interval 30

!
```

Verification

- Verify that the bundle status is UP.

```
router# show bundle bundle-ether 1200
Wed Dec 16 19:55:49.974 PST

Bundle-Ether12000
  Status:                                    Up
  Local links <active/standby/configured>:   35 / 0 / 35
  Local bandwidth <effective/available>:     3500000000 (3500000000) kbps
  MAC address (source):                      ea3b.745f.c4b0 (Chassis pool)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:                            Default
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                             Disabled
  mLACP:                                       Not configured
  IPv4 BFD:                                    Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Hu0/4/0/0             Local            Active       0x8000, 0x0009   100000000
      Link is Active
  Hu0/4/0/1             Local            Active       0x8000, 0x000a   100000000
      Link is Active
- - -
- - -
  Hu0/4/0/35            Local            Active       0x8000, 0x002b   100000000
      Link is Active
```

• Verify the bundle statistics:

```
router# show policy-map interface bundle-ether 12000

Bundle-Ether12000 input: ingress

Class inet4-classifier-af1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         4647401962/21236124455654       26403040
    Transmitted         :         4647401962/21236124455654       26403040
    Total Dropped       :                  0/0                    0
Class inet4-classifier-af2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         4502980177/20576584333939       25571493
    Transmitted         :         4502980177/20576584333939       25571493
    Total Dropped       :                  0/0                    0
Class inet4-classifier-af3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         4647404125/21236213667880       26389086
    Transmitted         :         4647404125/21236213667880       26389086
    Total Dropped       :                  0/0                    0
Class inet4-classifier-af4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         9291188840/42456120548683       52771168
    Transmitted         :         9291188840/42456120548683       52771168
    Total Dropped       :                  0/0                    0
Class inet4-classifier-be1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         4647413429/21235847852686       26393414
    Transmitted         :         4647413429/21235847852686       26393414
    Total Dropped       :                  0/0                    0
Class inet4-classifier-nc1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         9294887621/42473100149807       52778258
    Transmitted         :         9294887621/42473100149807       52778258
    Total Dropped       :                  0/0                    0

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                    0
    Transmitted         :                  0/0                    0
    Total Dropped       :                  0/0                    0

Bundle-Ether12000 output: egress

Class c1
  Classification statistics          (packets/bytes)     (rate - kbps)
```

```
    Matched            :        16665494532/75878118942463       8760591
    Transmitted        :        16655834643/75834136022017       8760591
    Total Dropped      :            9659889/43982920446          0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 9659889/43982920446
Class c2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :        16665421959/75877849543188       8718687
    Transmitted        :        16665421959/75877849543188       8718687
    Total Dropped      :                  0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
Class c3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :        16665247833/75877509455458       8703470
    Transmitted        :        16665187414/75877234624197       8703470
    Total Dropped      :              60419/274831261            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 60419/274831261
Class c4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :        33330896131/151755393012945      17470745
    Transmitted        :        33330745421/151754709368565      17470745
    Total Dropped      :             150710/683644380            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 150710/683644380
Class c5
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :        16878910340/76849791869834       8833394
    Transmitted        :        16878849464/76849514633309       8833394
    Total Dropped      :              60876/277236525            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 60876/277236525
Class c6
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :        33330898844/151756094112925      17456785
    Transmitted        :        33330752668/151755427708382      17456785
    Total Dropped      :             146176/666404543            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 146176/666404543
Class c7
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :             244106/79922040             74
    Transmitted        :             244106/79922040             74
    Total Dropped      :                  0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched            :       267075066180/1215993441123215     139917482
    Transmitted        :       267075066180/1215993441123215     139917482
    Total Dropped      :                  0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
```

Related Topics

- QoS on Link Bundles, on page 85