CHAPTER 12

Configuring Quality of Service on the ML-Series Card

This chapter describes the Quality of Service (QoS) features built into your ML-Series card. It also describes how to map QoS scheduling at both the system and interface levels.

This chapter contains the following major sections:

The ML-Series card employs the Cisco IOS Modular QoS Command-line Interface (MQC). For more information on general MQC configuration, refer to the following Cisco IOS documents:

- *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2* at this URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122mindx/l22index.htm
- *Cisco IOS Quality of Service Solutions Command Reference, Release 12.2* at this URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_r/index.htm

Understanding QoS

The ML-Series card multiplexes multiple IP/Ethernet services onto the SONET circuit and dynamically allocates transmission bandwidth to data services based on data service requirements. This allows the network to operate at a significantly higher level of utilization. To support service-level agreements (SLAs), this dynamic allocation must accommodate the service elements of bandwidth, including loss and delay. The characteristics of these service elements make up QoS.

The QoS mechanism has three basic steps. It classifies types of traffic, specifies what action to take against a type of traffic, and specifies where the action should take place.

Priority Mechanism in IP and Ethernet

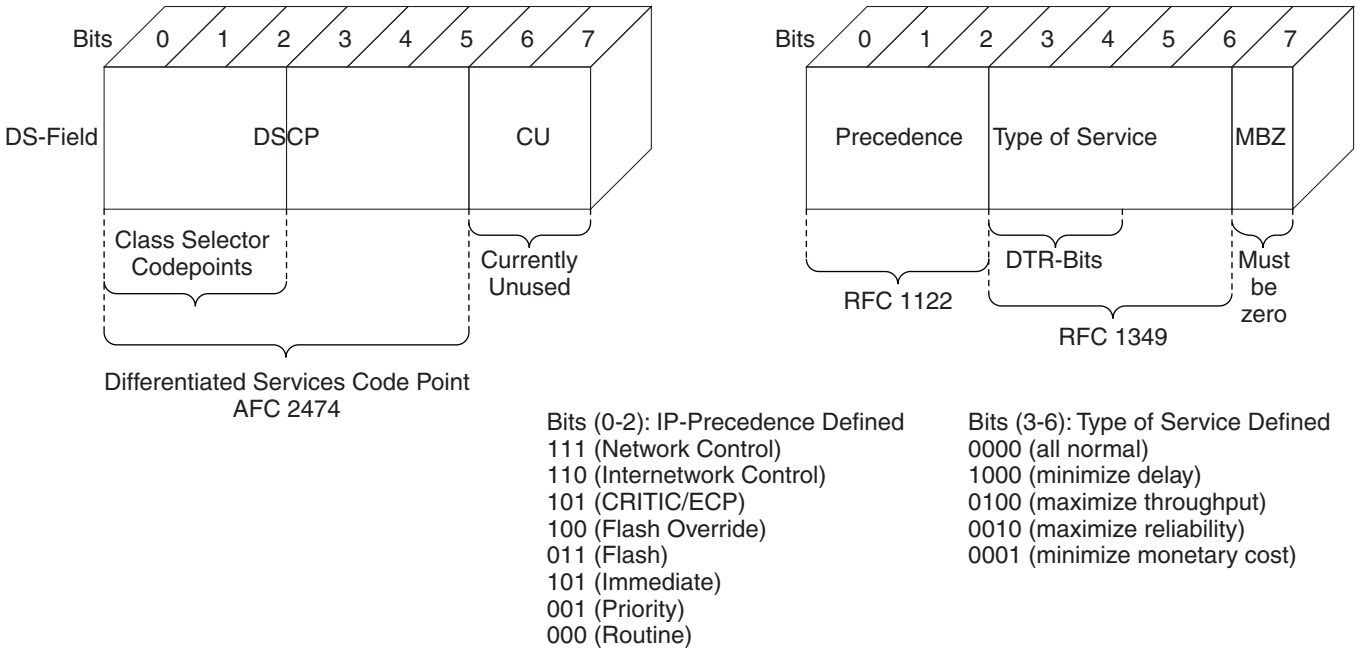
For any QoS service to be applied to data, there must be a way to mark or identify an IP packet or an Ethernet frame. When identified, a specific priority can be assigned to each individual IP packet or Ethernet frame. The IP Precedence or the IP Differentiated Services Code Point (DSCP) field prioritizes the IP packets, and the Ethernet class of service (IEEE 802.1p defined class of service [CoS]) is used for the Ethernet frames. IP precedence and Ethernet CoS are further described in the following sections.

IP Precedence and Differentiated Services Code Point

IP precedence uses the three precedence bits in the IPv4 header's ToS (type of service) field to specify class of service for each IP packet (RFC 1122). The most significant three bits of the IPv4 ToS field provide up to eight distinct classes, of which six are used for classifying services and the remaining two are reserved. On the edge of the network, the IP precedence is assigned by the client device or the ML Series, so that each subsequent network element can provide services based on the determined policy or the SLA.

IP DSCP uses the six bits in the IPv4 header to specify class of service for each IP packet (RFC 2474). Figure 12-1 illustrates IP precedence and DSCP. The DSCP field classifies packets into any of the 64 possible classes. On the network edge, the IP DSCP is assigned by the client device or the ML Series, so that each subsequent network element can provide services based on the determined policy or the SLA.

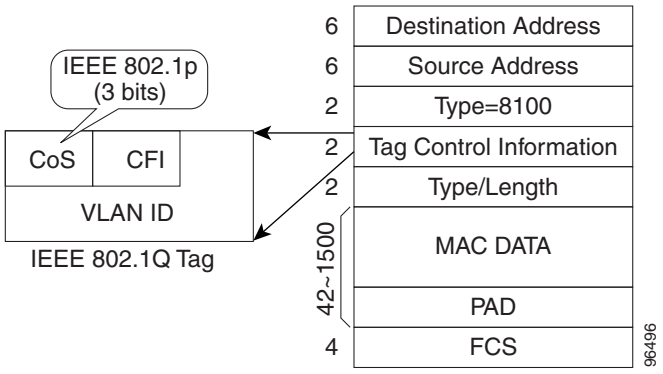
Figure 12-1 IP Precedence and DSCP



Ethernet CoS

Ethernet CoS refers to three bits within a four byte IEEE 802.1Q (VLAN) header used to indicate the priority of the Ethernet frame as it passes through a switched network. The CoS bits in the IEEE 802.1Q header are commonly referred to as the IEEE 802.1p bits. There are three CoS bits that provide eight classes, matching the number delivered by IP precedence. In many real-world networks, a packet might traverse both Layer 2 and Layer 3 domains. To maintain QoS across the network, the IP ToS can be mapped to the Ethernet CoS and vice versa, for example, in linear or one-to-one mapping, because each mechanism supports eight classes. Similarly, a set of DSCP values (64 classes) can be mapped into each of the eight individual Ethernet CoS values. Figure 12-2 is an IEEE 802.1Q Ethernet frame, which consists of a 2-byte Ethertype and a 2-byte tag (IEEE 802.1Q Tag) on the Ethernet protocol header.

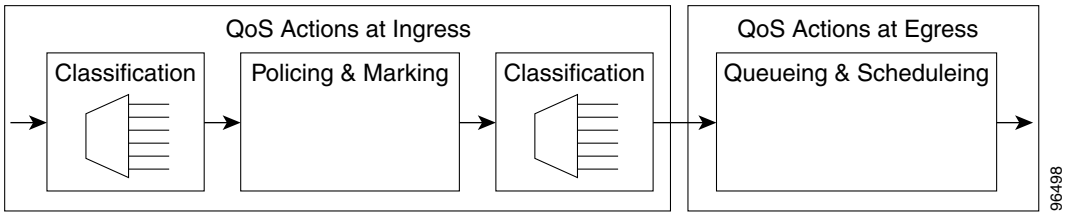
Figure 12-2 Ethernet Frame and the CoS Bit (IEEE 802.1p)



ML-Series QoS

The ML-Series QoS classifies each packet in the network based on its input interface, bridge group (VLAN), Ethernet CoS, IP precedence, IP DSCP, or resilient packet ring (RPR)-CoS. After they are classified into class flows, further QoS functions can be applied to each packet as it traverses the card. Figure 12-3 illustrates the ML-Series QoS flow.

Figure 12-3 ML-Series QoS Flow



Policing provided by the ML-Series card ensures that attached equipment does not submit more than a predefined amount of bandwidth (Rate Limiting) into the network. The policing feature can be used to enforce the committed information rate (CIR) and the peak information rate (PIR) available to a customer at an interface. Policing also helps characterize the statistical nature of the information allowed into the network so that traffic engineering can more effectively ensure that the amount of committed bandwidth is available on the network, and the peak bandwidth is over-subscribed with an appropriate ratio. The policing action is applied per classification.

Priority marking can set the Ethernet IEEE 802.1p CoS bits or RPR-CoS bits as they exit the ML-Series card. The marking feature operates on the outer IEEE 802.1p tag, and provides a mechanism for tagging packets at the ingress of a QinQ packet. The subsequent network elements can provide QoS based only on this service-provider-created QoS indicator.

Per-class flow queuing enables fair access to excess network bandwidth, allows allocation of bandwidth to support SLAs, and ensures that applications with high network resource requirements are adequately served. Buffers are allocated to queues dynamically from a shared resource pool. The allocation process incorporates the instantaneous system load as well as the allocated bandwidth to each queue to optimize buffer allocation. Congestion management on the ML-Series is performed through a tail drop mechanism along with discard eligibility on the egress scheduler.

The ML-Series uses a Weighted Deficit Round Robin (WDRR) scheduling process to provide fair access to excess bandwidth as well as guaranteed throughput to each class flow.

Admission control is a process that is invoked each time that service is configured on the ML-Series card to ensure that QoS resources are not overcommitted. In particular, admission control ensures that no configurations are accepted when the sum of committed bandwidths on an interface exceeds the total bandwidth on the interface.

Classification

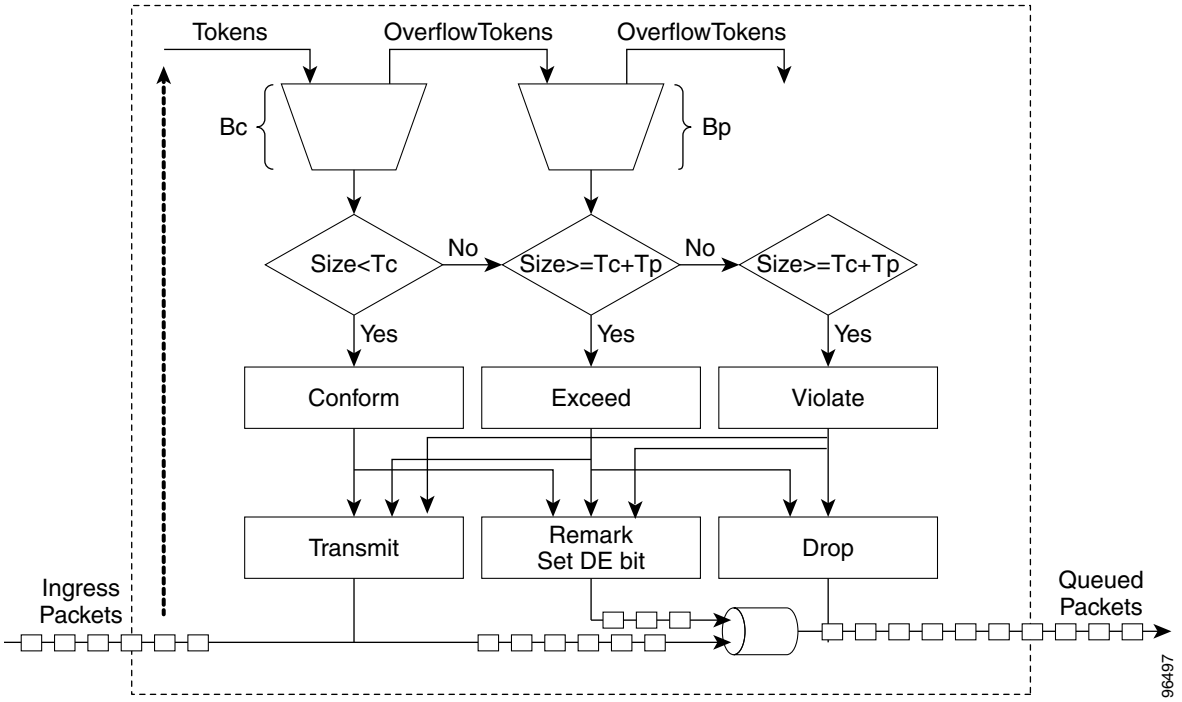
Classification can be based on any single packet classification criteria or a combination (logical AND and OR). Classification of packets is configured using the Modular CLI **class-map** command. For traffic transiting the RPR, only the input interface and/or the RPR-CoS can be used as classification criteria.

Policing

Dual leaky bucket policer is a process where the first bucket (CIR bucket) is filled with tokens at a known rate (CIR), which is a parameter that can be configured by the operator. Figure 12-4 illustrates the dual leaky bucket policer model. The tokens fill the bucket up to a maximum level, which is the amount of burstable committed (BC) traffic on the policer. The nonconforming packets of the first bucket are the overflow packets, which are passed to the second leaky bucket (the PIR bucket). The second leaky bucket is filled with these tokens at a known rate (PIR), which is a parameter that can be configured by the operator. The tokens fill the PIR bucket up to a maximum level (BP), which is the amount of peak burstable traffic on the policer. The nonconform packets of the second bucket are the overflow packets, which can be dropped or marked according to the policer definition.

On the dual leaky bucket policer, the packets conforming to the CIR are conform packets, the packets not conforming to CIR but conforming to PIR are exceed packets, and the packets not conforming to either the PIR or CIR are violate packets.

Figure 12-4 Dual Leaky Bucket Policer Model



Marking and Discarding with a Policer

On the ML-Series card's policer, the conform packets can be transmitted or marked and transmitted. The exceed packets can be transmitted, marked and transmitted, or dropped. The violating packets can be transmitted, marked and transmitted, or dropped. The primary application of the dual-rate or three-color policer is to mark the conform packets with CoS bit 2l, mark the exceed packet with CoS bit 1, and discard the violated packets so all the subsequent network devices can implement the proper QoS treatment per frame/packet basis based on these priority marking without knowledge of each SLA.

In some cases, it might be desirable to discard all traffic of a specific ingress class. This can be accomplished by using a police command of the following form with the class: **police 96000 conform-action drop exceed-action drop**.

If a marked packet has a provider-supplied Q-tag inserted before transmission, the marking only affects the provider Q-tag. If a Q-tag is received, it is re-marked. If a marked packet is transported over the RPR ring, the marking also affects the RPR-CoS bit.

If a Q-tag is inserted (QinQ), the marking affects the added Q-tag. If the ingress packet contains a Q-tag and is transparently switched, the existing Q-tag is marked. In case of a packet without any Q-tag, the marking does not have any significance.

The local scheduler treats all nonconforming packets as discard eligible regardless of their CoS setting or the global cos commit definition. For RPR implementation, the discard eligible (DE) packets are marked using the DE bit on the RPR header. The discard eligibility based on the CoS commit or the policing action is local to the ML-Series card scheduler, but it is global for the RPR ring.

Queuing

ML-Series card queuing uses a shared buffer pool to allocate memory dynamically to different traffic queues. The ML-100T-8 has 1.5 MB of packet buffer memory.

Each queue has an upper limit on the allocated number of buffers based on the class bandwidth assignment of the queue and the number of queues configured. This upper limit is typically 30 percent to 50 percent of the shared buffer capacity. Dynamic buffer allocation to each queue can be reduced based on the number of queues needing extra buffering. The dynamic allocation mechanism provides fairness in proportion to service commitments as well as optimization of system throughput over a range of system traffic loads.

The Low Latency Queue (LLQ) is defined by setting the weight to infinity or committing 100 percent bandwidth. When a LLQ is defined, a policer should also be defined on the ingress for that specific class to limit the maximum bandwidth consumed by the LLQ; otherwise there is a potential risk of LLQ occupying the whole bandwidth and starving the other unicast queues.

The ML-Series includes support for 400 user-definable queues, which are assigned per the classification and bandwidth allocation definition. The classification used for scheduling classifies the frames/packet after the policing action, so if the policer is used to mark or change the CoS bits of the ingress frames/packet, the new values are applicable for the classification of traffic for queuing and scheduling. The ML-Series provides buffering for 4000 packets.

Scheduling

Scheduling is provided by a series of schedulers that perform a WDRR as well as priority scheduling mechanisms from the queued traffic associated with each egress port.

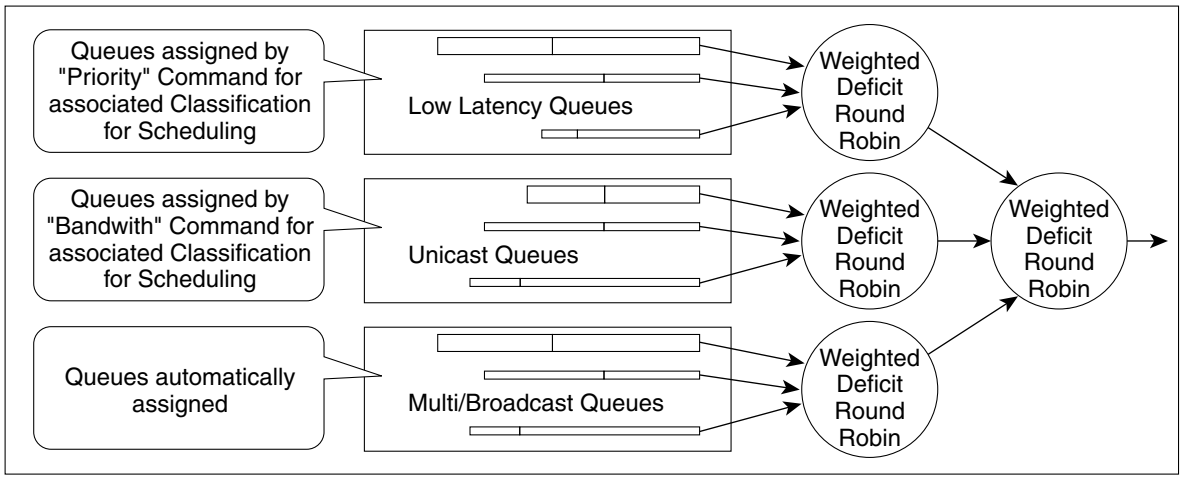
Though ordinary round robin servicing of queues can be done in constant time, unfairness occurs when different queues use different packet sizes. Deficit Round Robin (DRR) scheduling solves this problem. If a queue was not able to send a packet in its previous round because its packet size was too large, the remainder from the previous amount of credits that the queue got in each previous round (quantum) is added to the quantum for the next round.

WDRR extends the quantum idea from the DRR to provide weighted throughput for each queue. Different queues have different weights, and the quantum assigned to each queue in its round is proportional to the relative weight of the queue among all the queues serviced by that scheduler.

Weights are assigned to each queue as a result of the service provisioning process. When coupled with policing and policy mapping provisioning, these weights and the WDRR scheduling process ensure that QoS commitments are provided to each service flow.

Figure 12-5 illustrates the ML-Series card's queuing and scheduling.

Figure 12-5 Queuing and Scheduling Model



The weighting structure allows traffic to be scheduled at 1/2048 of the port rate. This equates to approximately 49 kbps for traffic exiting a FastEthernet port.

The unicast queues are created as the output service policy implementation on the egress ports. Each unicast queue is assigned with a committed bandwidth and the weight of the queue is determined by the normalization of committed bandwidth of all defined unicast queues for that port. The traffic beyond the committed bandwidth on any queue is treated by the scheduler according to the relative weight of the queue.

The LLQ is created as the output service policy implementation on the egress ports. Each LLQ is assigned with a committed bandwidth of 100 percent and is served with lower latency. To limit the bandwidth usage by the LLQ, a strict policer needs to be implemented on the ingress for the LLQ traffic classes.

The DE allows some packets to be treated as committed and some as discard-eligible on the scheduler. For the Ethernet frames, the CoS (IEEE 802.1p) bits are used to identify committed and discard eligible packets, where the RPR-CoS and the DE bits are used for RPR traffic. When congestion occurs and a queue begins to fill, the DE packets hit a lower tail-drop threshold than the committed packets. Committed packets are not dropped until the total committed load exceeds the interface output. The tail-drop thresholds adjust dynamically in the card to maximize use of the shared buffer pool while guaranteeing fairness under all conditions.

Control Packets and L2 Tunneled Protocols

The control packets originated by the ML-Series card have a higher priority than data packets. The external Layer 2 and Layer 3 control packets are handled as data packets and assigned to broadcast queues. Bridge protocol data unit (BPDU) prioritization in the ML-Series card gives Layer 2-tunneled BPDU sent out the multicast/broadcast queue a higher discard value and therefore a higher priority than than other packets in the multicast/broadcast queue. The Ethernet CoS (IEEE 802.1p) for Layer 2-tunneled protocols can be assigned by the ML-Series card.

Egress Priority Marking

Egress priority marking allows the operator to assign the IEEE 802.1p CoS bits of packets that exit the card. This marking allows the operator to use the CoS bits as a mechanism for signaling to downstream nodes the QoS treatment that the packet should be given. This feature operates on the outer-most IEEE 802.1p CoS field. When used with the QinQ feature, priority marking allows the user traffic (inner Q-tag) to traverse the network transparently, while providing a means for the network to internally signal QoS treatment at Layer 2.

Priority marking follows the classification process, and therefore any of the classification criteria identified earlier can be used as the basis to set the outgoing IEEE 802.1p CoS field. For example, a specific CoS value can be mapped to a specific bridge group.

Priority marking is configured using the MQC **set-cos** command. If packets would otherwise leave the card without an IEEE 802.1Q tag, then the **set-cos** command has no effect on that packet. If an IEEE 802.1Q tag is inserted in the packet (either a normal tag or a QinQ tag), the inserted tag has the set-cos priority. If an IEEE 802.1Q tag is present on packet ingress and retained on packet egress, the priority of that tag is modified. If the ingress interface is an QinQ access port and the **set-cos** policy-map classifies based on ingress tag priority, this classifies based on the user priority. This is a way to allow the user-tag priority to determine the SP tag priority. When a packet does not match any **set-cos** policy-map, the priority of any preserved tag is unchanged and the priority of any inserted IEEE 802.1Q tag is set to 0.

The **set-cos** command on the output service policy is only applied to unicast traffic. Priority marking for multicast/broadcast traffic can only be achieved by the **set-cos** action of the policing process on the input service policy.

Ingress Priority Marking

Ingress priority marking can be done for all input packets of a port, for all input packets matching a classification, or based on a measured rate. Marking of all packets of an input class can also be done with a policing command of the form **police 96000 conform-action set-cos-transmit exceed-action set-cos-transmit**. Using this command with a policy map that contains only the “class-default” will mark all ingress packets to the value. Rate based priority marking is discussed in the “Marking and Discarding with a Policer” section on page 12-5.

QinQ Implementation

The hierarchical VLAN or IEEE 802.1Q tunneling feature enables the service provider to transparently carry the customer VLANs coming from any specific port (UNI) and transport them over the service provider network. This feature is also known as QinQ, which is performed by adding an additional IEEE 802.1Q tag on every customer frame.

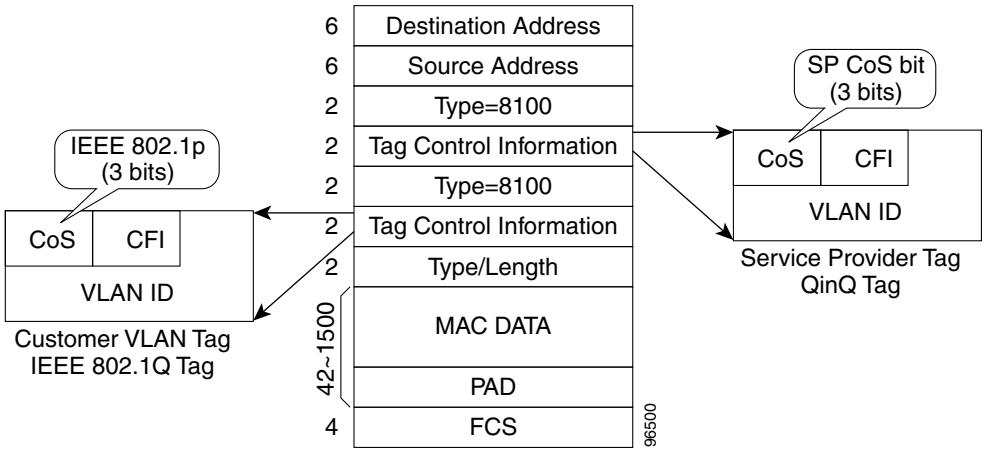
Using the QinQ feature, service providers can use a single VLAN to support customers with multiple VLANs. QinQ preserves customer VLAN IDs and segregates traffic from different customers within the service-provider infrastructure, even when traffic from different customers originally shared the same VLAN ID. The QinQ also expands VLAN space by using a VLAN-in-VLAN hierarchy and tagging the tagged packets. When the service provider (SP) tag is added, the QinQ network typically loses any visibility to the IP header or the customer Ethernet IEEE 802.1Q tag on the QinQ encapsulated frames.

On the ML-Series cards, the QinQ access ports (IEEE 802.1Q tunnel ports or QinQ UNI ports) have visibility to the customer CoS and the IP precedence or IP DSCP values; therefore, the SP tag can be assigned with proper CoS bit, which would reflect the customer IP precedence, IP DSCP, or CoS bits. In

the QinQ network, the QoS is then implemented based on the IEEE 802.1p bit of the SP tag. The ML-Series cards do not have visibility into the customer CoS, IP precedence, or DSCP values after the packet is double-tagged (because it is beyond the entry point of the QinQ service).

Figure 12-6 illustrates the QinQ implementation on the ML-Series card.

Figure 12-6 QinQ Implementation on the ML-Series Card



The ML-Series cards can be used as the IEEE 802.1Q tunneling device for the QinQ network and also provide the option to copy the customer frame's CoS bit into the CoS bit of the added QinQ tag. This allows the service provider QinQ network to be fully aware of the necessary QoS treatment for each individual customer frame.

Flow Control Pause and QoS

If flow control and port-based policing are both enabled for an interface, flow control handles the bandwidth. If the policer gets noncompliant flow, then the policer drops or demarks the packets using the policer definition of the interface.

Note QoS and policing are not supported on the ML-Series card interface when link aggregation is used.

Note Egress shaping is not supported on the ML-Series cards.

QoS on RPR

For VLAN bridging over RPR, all ML-Series cards on the ring must be configured with the base RPR and RPR QoS configuration. SLA and bridging configurations are only needed at customer RPR access points, where IEEE 802.1Q VLAN CoS is copied to the RPR CoS. This IEEE 802.1Q VLAN CoS copying can be overwritten with a **set-cos** *action* command. The CoS commit rule applies at the RPR ring ingress. Transit RPR ring traffic is classified on CoS only.

If the packet does not have a VLAN header, the RPR CoS for non-VLAN traffic is set using the following rules:

1. The default CoS is 0.

2. If the packet comes in with an assigned CoS, the assigned CoS replaces the default. If an IP packet originates locally, the IP precedence setting replaces the CoS setting.
3. The input policy map has a **set-cos** action.
4. The output policy map has a **set-cos** action (except for broadcast or multicast packets).

The RPR header contains a CoS value and DE indicator. The RPR DE is set for noncommitted traffic.

Configuring QoS

This section describes the tasks for configuring the ML-Series card QoS functions using the MQC. The ML-Series card does not support the full set of MQC functionality.

To configure and enable class-based QoS features, perform the procedures described in the following sections:

- Creating a Traffic Class, page 12-10
- Creating a Traffic Policy, page 12-11
- Attaching a Traffic Policy to an Interface, page 12-15
- Configuring CoS-Based QoS, page 12-16

For QoS configuration examples, see the "QoS Configuration Examples" section on page 12-17.

Creating a Traffic Class

The **class-map** global configuration command is used to create a traffic class. The syntax of the **class-map** command is as follows:

class-map [**match-any** | **match-all**] *class-map-name*
no class-map [**match-any** | **match-all**] *class-map-name*

The match-all and match-any options need to be specified only if more than one match criterion is configured in the traffic class. The **class-map match-all** command is used when all of the match criteria in the traffic class must be met for a packet to match the specified traffic class. The **class-map match-any** command is used when only one of the match criterion in the traffic class must be met for a packet to match the specified traffic class. If neither the **match-all** nor **match-any** keyword is specified, the traffic class behaves in a manner consistent with **class-map match-all** command.

To create a traffic class containing match criteria, use the **class-map** global configuration command to specify the traffic class name, and then use the **match** commands in Table 12-1, as needed.

Table 12-1 Traffic Class Commands

Command	Purpose
`ML_Series(config)# class-map class-map-name`	Specifies the user-defined name of the traffic class. Names can be a maximum of 40 alphanumeric characters. If **match-all** or **match-any** is not specified, traffic must match all the match criteria to be classified as part of the traffic class. There is no default-match criteria. Multiple match criteria are supported. The command matches either all or any of the criteria, as controlled by the **match-all** and **match-any** subcommands of the **class-map** command. **Note** The ML-100T-8 supports a maximum of 126 user-defined class maps, plus one default class map named "class-default". The ML-Series card on the ONS 15454 SONET/SDH supports a maximum of 254 user-defined class maps, plus one default class map named "class-default".
`ML_Series(config)# class-map match-all class-map-name`	Specifies that all match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
`ML_Series(config)# class-map match-any class-map-name`	Specifies that one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
`ML_Series(config-cmap)# match any`	Specifies that all packets will be matched.
`ML_Series(config-cmap)# match bridge-group bridge-group-number`	Specifies the bridge-group-number against whose contents packets are checked to determine if they belong to the class.
`ML_Series(config-cmap)# match cos cos-number`	Specifies the CoS value against whose contents packets are checked to determine if they belong to the class.
`ML_Series(config-cmap)# match input-interface interface-name`	Specifies the name of the input interface used as a match criterion against which packets are checked to determine if they belong to the class. The shared packet ring (SPR) interface used in RPR (SPR1) is a valid interface-name for the ML-Series card. For more information on the SPR interface, see Chapter 15, "Configuring Resilient Packet Ring on the ML-Series Card." The **input-interface** choice is not valid when applied to the INPUT of an interface (redundant).
`ML_Series(config-cmap)# match ip dscp ip-dscp-value`	Specifies up to eight DSCP values used as match criteria. The value of each service code point is from 0 to 63.
`ML_Series (config-cmap)# match ip precedence ip-precedence-value`	Specifies up to eight IP precedence values used as match criteria.

Creating a Traffic Policy

To configure a traffic policy, use the **policy-map** global configuration command to specify the traffic policy name, and use the following configuration commands to associate a traffic class, which was configured with the **class-map** command and one or more QoS features. The traffic class is associated

with the traffic policy when the **class** command is used. The **class** command must be issued after entering policy-map configuration mode. After entering the **class** command, you are automatically in policy-map class configuration mode, which is where the QoS policies for the traffic policy are defined.

When the bandwidth or priority action is used on any class in a policy map, then there must be a class, defined by the **match-any** command, that has a bandwidth or priority action in that policy map. This is to ensure that all traffic can be classified into a default class that has some assigned bandwidth. A minimum bandwidth can be assigned if the class is not expected to be used or if no reserved bandwidth is desired for default traffic.

The QoS policies that can be applied in the traffic policy in policy-map class configuration mode are detailed in the following example.

The syntax of the **policy-map** command is:

policy-map *policy-name*
no policy-map *policy-name*

The syntax of the **class** command is:

class *class-map-name*
no class *class-map-name*

All traffic that fails to meet the matching criteria belongs to the default traffic class. The default traffic class can be configured by the user, but cannot be deleted.

To create a traffic policy, use the commands in Table 12-2 as needed.

Table 12-2 Traffic Policy Commands

Command	Purpose
`ML_Series (config)# policy-map policy-name`	Specifies the name of the traffic policy to configure. Names can be a maximum of 40 alphanumeric characters.
`ML_Series (config-pmap)# class class-map-name`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.
`ML_Series (config-pmap)# class class-default`	Specifies the default class to be created as part of the traffic policy.

Table 12-2 Traffic Policy Commands (continued)

Command	Purpose
`ML_Series (config-pmap-c)# bandwidth {bandwidth-kbps | percent percent}`	Specifies a minimum bandwidth guarantee to a traffic class in periods of congestion. A minimum bandwidth guarantee can be specified in kbps or by a percentage of the overall available bandwidth. Valid choices for the ML-Series cards are: • Rate in kilobits per second • Percent of total available bandwidth (1 to 100) If multiple classes and bandwidth actions are specified in a single policy map, they must use the same choice in specifying bandwidth (kilobits or percent). **Note** When using the **bandwidth** command, excess traffic (beyond the configured commit) is allocated any available bandwidth in proportion to the relative bandwidth commitment of its traffic class compared to other traffic classes. Excess traffic from two classes with equal commits has equal access to available bandwidth. Excess traffic from a class with a minimum commit might receive only a minimum share of available bandwidth compared to excess bandwidth from a class with a high commit. **Note** The true configurable bandwidth in kilobits per second is per port and depends on how the ML-Series card is configured. The **show interface** command shows the maximum bandwidth of a port (for example, BW 100000 Kbit). The sum of all bandwidth and priority actions applied to the interface, plus the cos priority-mcast bandwidth, is not allowed to exceed the maximum bandwidth of the port.

Table 12-2 Traffic Policy Commands (continued)

Command	Purpose
`Router (config-pmap-c)# police` *cir-rate-bps normal-burst-byte* [*max-burst-byte*] [**pir** *pir-rate-bps*] [**conform-action {set-cos-transmit | transmit | drop}] [exceed-action {set-cos-transmit | drop}]** [**violate-action {set-cos-transmit | drop}]**	Defines a policer for the currently selected class when the policy map is applied to input. Policing is supported only on ingress, not on egress. • For *cir-rate-bps,* specify the average committed information rate (cir) in bits per second (bps). The range is 96000 to 800000000. • For *normal-burst-byte,* specify the cir burst size in bytes. The range is 8000 to 64000. • (Optional) For *maximum-burst-byte*, specify the peak information rate (pir) burst in bytes. The range is 8000 to 64000. • (Optional) For *pir-rate-bps*, specify the average pir traffic rate in bps where the range is 96000 to 800000000. • (Optional) Conform action options are: – Set a CoS priority value and transmit – Transmit packet (default) – Drop packet • (Optional) Exceed action options are: – Set a CoS value and transmit – Drop packet (default) • (Optional) The violate action is only valid if pir is configured. Violate action options are: – Set a CoS value and transmit – Drop packet (default)

Table 12-2 Traffic Policy Commands (continued)

Command	Purpose
`ML_Series (config-pmap-c)# priority kbps`	Specifies low latency queuing for the currently selected class. This command can only be applied to an output. When the policy-map is applied to an output, an output queue with strict priority is created for this class. The only valid rate choice is in kilobits per second. **Note** This **priority** command does not apply to the default class. **Note** When using the priority action, the traffic in that class is given a 100 percent CIR, regardless of the rate entered as the priority rate. To ensure that other bandwidth commitments are met for the interface, a policer must be configured on the input of all interfaces that might deliver traffic to this output class, limiting the peak rate to the priority rate entered. **Note** The true configurable bandwidth in kilobits per second is per port and depends on how the ML-Series card is configured. The **show interface** command shows the maximum bandwidth of a port (for example, BW 100000 Kbit). The sum of all bandwidth and priority actions applied to the interface, plus the cos priority-mcast bandwidth, is not allowed to exceed the maximum bandwidth of the port.
`ML_Series (config-pmap-c)# set cos cos-value`	Specifies a CoS value or values to associate with the packet. The number is in the range from 0 to 7. This command can only be used in a policy-map applied to an output. It specifies the VLAN CoS priority to set for the outbound packets in the currently selected class. If QinQ is used, the top-level VLAN tag is marked. If outbound packets have no VLAN tag, the action has no effect. This action is applied to the packet after any set-cos action done by a policer, and therefore overrides the CoS set by a policer action. If a packet is marked by the policer and forwarded out through an interface that also has a set-cos action assigned for the traffic class, the value specified by the police action takes precedence in setting the IEEE 802.1p CoS field. This command also sets the CoS value in the RPR header for packets exiting the ML-Series on the RPR interface.

Attaching a Traffic Policy to an Interface

Use the **service-policy** interface configuration command to attach a traffic policy to an interface and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface). Only one traffic policy can be applied to an interface in a given direction.

Use the **no** form of the command to detach a traffic policy from an interface. The **service-policy** command syntax is as follows:

service-policy {**input** | **output**} *policy-map-name*
no service-policy {**input** | **output**} *policy-map-name*

To attach a traffic policy to an interface, perform the following procedure in global configuration mode:

	Command	Purpose
Step 1	`ML_Series(config)# interface interface-id`	Enters interface configuration mode, and specifies the interface to apply the policy map. Valid interfaces are limited to physical Ethernet and packet-over-SONET (POS) interfaces. **Note** Policy maps cannot be applied to SPR interfaces, subinterfaces, port channel interfaces, or Bridge Group Virtual Interfaces (BVIs).
Step 2	`ML_Series(config-if)# service-policy output policy-map-name`	Specifies the name of the traffic policy to be attached to the output direction of an interface. The traffic policy evaluates all traffic leaving that interface.
Step 3	`ML_Series(config-if)# service-policy input policy-map-name`	Specifies the name of the traffic policy to be attached to the input direction of an interface. The traffic policy evaluates all traffic entering that interface.

Configuring CoS-Based QoS

The global **cos commit** *cos-value* command allows the ML-Series card to base the QoS treatment for a packet coming in on a network interface on the attached CoS value, rather than on a per-customer-queue policer.

CoS-based QoS is applied with a single global **cos commit** *cos-value* command, as shown in Table 12-3:

Table 12-3 CoS Commit Command

Command	Purpose
`ML_Series(config)# cos-commit cos-value`	Labels packets that come in with a CoS equal to or higher than the *cos-value* as CIR and packets with a lower CoS as DE.

Monitoring and Verifying QoS Configuration

After configuring QoS on the ML-Series card, the configuration of class maps and policy maps can be viewed through a variety of **show** commands. To display the information relating to a traffic class or traffic policy, use one of the following commands in EXEC mode, as needed. Table 12-4 describes the commands that are related to QoS status.

Table 12-4 Commands for QoS Status

Command	Purpose
`ML_Series# show class-map name`	Displays the traffic class information of the user-specified traffic class.
`ML_Series# show policy-map`	Displays all configured traffic policies.

Table 12-4 Commands for QoS Status (continued)

Command	Purpose
ML_Series# **show policy-map** *name*	Displays the user-specified policy map.
ML_Series# **show policy-map interface** *interface*	Displays configurations of all input and output policies attached to an interface. Statistics displayed with this command are unsupported and show zero.

Example 12-1 shows examples of the QoS commands.

Example 12-1 QoS Status Command Examples

```
ML_Series# show class-map
 Class Map match-any class-default (id 0)
   Match any
 Class Map match-all policer (id 2)
   Match ip precedence 0

ML_Series# show policy-map
 Policy Map police_f0
  class  policer
   police 1000000 10000 conform-action transmit exceed-action drop

ML_Series# show policy-map interface

 FastEthernet0

  service-policy input: police_f0

    class-map: policer (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      match: ip precedence 0

    class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      match: any
        0 packets, 0 bytes
        5 minute rate 0 bps
```

QoS Configuration Examples

This section provides the specific command and network configuration examples:

- Traffic Classes Defined Example
- Traffic Policy Created Example
- class-map match-any and class-map match-all Commands Example
- match spr1 Interface Example
- ML-Series VoIP Example
- ML-Series Policing Example
- ML-Series CoS-Based QoS Example

Traffic Classes Defined Example

Example 12-2 shows how to create a class map called class1 that matches incoming traffic entering interface fastethernet0.

Example 12-2 Class Interface Command Example

```
ML_Series(config)# class-map class1
ML_Series(config-cmap)# match input-interface fastethernet0
```

Example 12-3 shows how to create a class map called class2 that matches incoming traffic with IP-precedence values of 5, 6, and 7.

Example 12-3 Class IP-Precedence Command Example

```
ML_Series(config)# class-map match-any class2
ML_Series(config-cmap)# match ip precedence 5 6 7
```

Note If a class-map contains a match rule that specifies multiple values, such as 5 6 7 in this example, then the class-map must be match-any, not the default match-all. Without the match-any class-map, an error message is printed and the class is ignored. The supported commands that allow multiple values are **match cos**, **match ip precedence**, and **match ip dscp**.

Example 12-4 shows how to create a class map called class3 that matches incoming traffic based on bridge group 1.

Example 12-4 Class Map Bridge Group Command Example

```
ML_Series(config)# class-map class3
ML_Series(config-cmap)# match bridge-group 1
```

Traffic Policy Created Example

In Example 12-5, a traffic policy called policy1 is defined to contain policy specifications, including a bandwidth allocation request for the default class and two additional classes—class1 and class2. The match criteria for these classes were defined in the traffic classes (see the "Creating a Traffic Class" section on page 12-10).

Example 12-5 Traffic Policy Created Example

```
ML_Series(config)# policy-map policy1
ML_Series(config-pmap)# class class-default
ML_Series(config-pmap-c)# bandwidth 1000
ML_Series(config-pmap)# exit

ML_Series(config-pmap)# class class1
ML_Series(config-pmap-c)# bandwidth 3000
ML_Series(config-pmap)# exit

ML_Series(config-pmap)# class class2
ML_Series(config-pmap-c)# bandwidth 2000
ML_Series(config-pmap)# exit
```

class-map match-any and class-map match-all Commands Example

This section illustrates the difference between the **class-map match-any** command and the **class-map match-all** command. The **match-any** and **match-all** options determine how packets are evaluated when multiple match criteria exist. Packets must either meet all of the match criteria (**match-all**) or one of the match criteria (**match-any**) in order to be considered a member of the traffic class.

Example 12-6 shows a traffic class configured with the **class-map match-all** command.

Example 12-6 Class Map Match All Command Example

```
ML_Series(config)# class-map match-all cisco1
ML_Series(config-cmap)# match cos 1
ML_Series(config-cmap)# match bridge-group 10
```

If a packet arrives with a traffic class called cisco1 configured on the interface, the packet is evaluated to determine if it matches the cos 1 and bridge group 10. If both of these match criteria are met, the packet matches traffic class cisco1.

In a traffic class called cisco2, the match criteria are evaluated consecutively until a successful match criterion is located. The packet is first evaluated to the determine whether cos 1 can be used as a match criterion. If cos 1 can be used as a match criterion, the packet is matched to traffic class cisco2. If cos 1 is not a successful match criterion, then bridge-group 10 is evaluated as a match criterion. Each matching criterion is evaluated to see if the packet matches that criterion. When a successful match occurs, the packet is classified as a member of traffic class cisco2. If the packet matches none of the specified criteria, the packet is classified as a member of the traffic class.

Note that the **class-map match-all** command requires that all of the match criteria must be met in order for the packet to be considered a member of the specified traffic class (a logical AND operator). In the example, cos 1 AND bridge group 10 have to be successful match criteria. However, only one match criterion must be met for the packet in the **class-map match-any** command to be classified as a member of the traffic class (a logical OR operator).

Example 12-7 shows a traffic class configured with the **class-map match-any** command. In the example, cos 1 OR bridge group 10 OR ip dscp 5 has to be successful match criteria.

Example 12-7 Class Map Match Any Command Example

```
ML_Series(config)# class-map match-any cisco2
ML_Series(config-cmap)# match cos 1
ML_Series(config-cmap)# match bridge-group 10
ML_Series(config-cmap)# match ip dscp 5
```

match spr1 Interface Example

In Example 12-8, the SPR interface is specified as a parameter to the **match input-interface** CLI when defining a class-map.

Example 12-8 Class Map SPR Interface Command Example

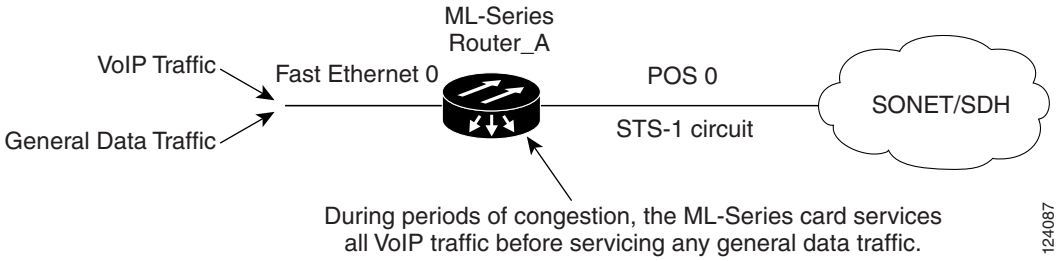
```
ML_Series(config)# class-map spr1-cos1
ML_Series(config-cmap)# match input-interface spr1
ML_Series(config-cmap)# match cos 1
ML_Series(config-cmap)# end
ML_Series# sh class-map spr1-cos1
 Class Map match-all spr1-cos1 (id 3)
```

```
Match input-interface SPR1
Match cos  1
```

ML-Series VoIP Example

Figure 12-7 shows an example of ML-Series voice-over- IP (VoIP). The associated commands are provided in Example 12-9.

Figure 12-7 ML-Series VoIP Example



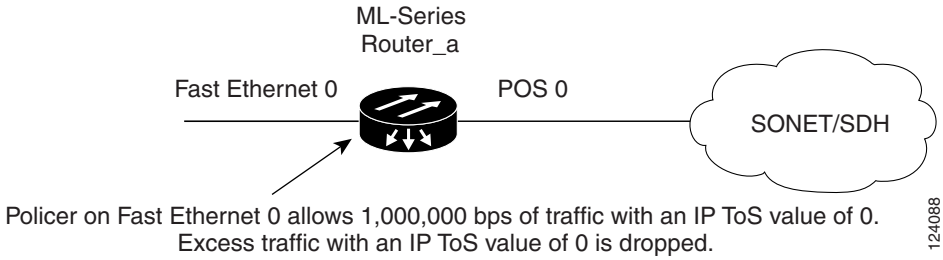
Example 12-9 ML-Series VoIP Commands

```
Router(config)# class-map match-all voip
Router(config-cmap)# match ip precedence 5
Router(config-cmap)# exit
Router(config)# class-map match-any default
Router(config-cmap)# match any
Router(config-cmap)# exit
Router(config)# policy-map pos0
Router(config-pmap)# class default
Router(config-pmap-c)# bandwidth   1000
Router(config-pmap-c)# class voip
Router(config-pmap-c)# priority 1000
Router(config-pmap-c)# interface FastEthernet0
Router(config-if)# ip address 1.1.1.1 255.255.255.0
Router(config-if)# interface POS0
Router(config-if)# ip address 2.1.1.1 255.255.255.0
Router(config-if)# service-policy output pos0
Router(config-if)# crc 32
Router(config-if)# no cdp enable
```

ML-Series Policing Example

Figure 12-8 shows an example of ML-Series policing. The example shows how to configure a policer that restricts traffic with an IP precedence of 0 to 1,000,000 bps. The associated code is provided in Example 12-10.

Figure 12-8 ML-Series Policing Example



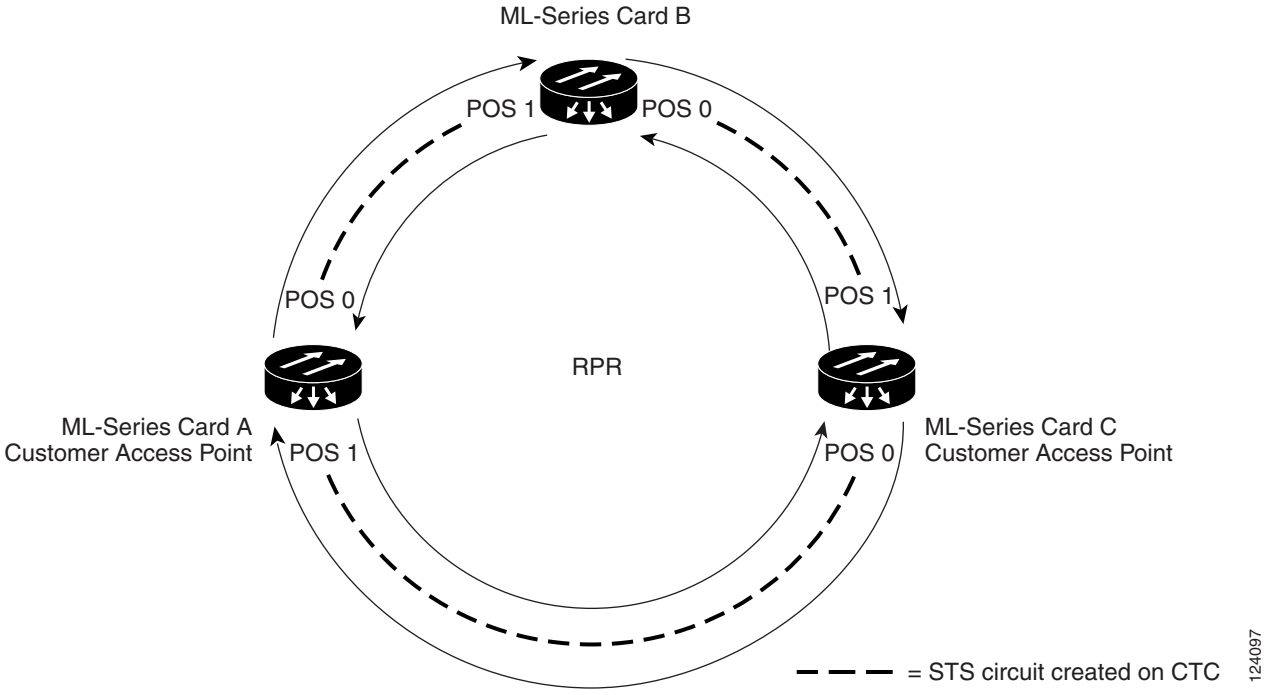
Example 12-10 ML-Series Policing Commands

```
Router(config)# class-map match-all policer
Router(config-cmap)# match ip precedence 0
Router(config-cmap)# exit
Router(config)# policy-map police_f0
Router(config-pmap)# class policer
Router(config-pmap-c)# police 1000000 10000 conform-action transmit exceed-action drop
Router(config-pmap-c)# interface FastEthernet0
Router(config-if)# service-policy input police_f0
```

ML-Series CoS-Based QoS Example

Figure 12-9 shows an example of ML-Series CoS-based QoS. The associated code is provided in the examples that follow the figure. The CoS example assumes that the ML-Series cards are configured into an RPR and that the ML-Series card POS ports are linked by point-to-point SONET circuits. ML-Series Card A and ML-Series Card C are customer access points. ML-Series Card B is not a customer access point. For more information on configuring RPR, see Chapter 15, "Configuring Resilient Packet Ring on the ML-Series Card."

Figure 12-9 ML-Series CoS Example



Example 12-11 shows the code used to configure ML-Series Card A in Figure 12-9.

Example 12-11 ML-Series Card A Configuration (Customer Access Point)

```
ML_Series_A(config)# cos commit 2
ML_Series_A(config)# policy-map Fast5_in
ML_Series_A(config-pmap)# class class-default
ML_Series_A(config-pmap-c)# police 5000 8000 8000 pir 10000 conform-action
set-cos-transmit 2 exceed-action set-cos-transmit 1 violate-action drop
```

Example 12-12 shows the code used to configure ML-Series Card B in Figure 12-9.

Example 12-12 ML-Series Card B Configuration (Not a Customer Access Point)

```
ML_Series_B(config)# cos commit 2
```

Example 12-13 shows the code used to configure ML-Series Card C in Figure 12-9.

Example 12-13 ML-Series Card C Configuration (Customer Access Point)

```
ML_Series_B(config)# cos commit 2
ML_Series_B(config)# policy-map Fast5_in
ML_Series_B(config-pmap)# class class-default
ML_Series_B(config-pmap-c)# police 5000 8000 8000 pir 10000 conform-action
set-cos-transmit 2 exceed-action set-cos-transmit 1 violate-action drop
```

Understanding Multicast QoS and Multicast Priority Queuing

ML-Series card QoS supports the creation of two priority classes for multicast traffic in addition to the default multiclass traffic class. Creating a multicast priority queuing class of traffic configures the ML-Series card to recognize an existing CoS value in ingressing multicast traffic for priority treatment.

The multicast priority queuing CoS match is based on the "internal" CoS value of each packet. This value is normally the same as the egress CoS value (after policer marking if enabled) but differs in two cases. The "internal" CoS value is not the same as the egress value when dot1q-tunneling is used. With dot1q-tunneling, the internal CoS value is always the value of the outer tag CoS, both when entering the dot1q tunnel and leaving the dot1q tunnel. The "internal" CoS value is also not the same as the egress value if a packet is transported over a VLAN, and the VLAN tag is removed on egress to send the packet untagged. In this case, the internal CoS is the CoS of the removed tag (including ingress policing and marking if enabled).

The **cos priority-mcast** command does not modify the CoS of the multicast packets but only the bandwidth allocation for the multicast priority queuing class. The command guarantees a minimum amount of bandwidth and is queued separately from the default multicast/broadcast queue.

Creating a multicast priority queuing class allows for special handling of certain types of multiclass traffic. This is especially valuable for multicast video distribution and service provider multicast traffic. For example, a service provider might want to guarantee the protection of their own multicast management traffic. To do this, they could create a multicast priority queuing class on the ML-Series card for the CoS value of the multicast management traffic and guarantee its minimum bandwidth. For multicast video distribution, a multicast priority queuing class on the ML-Series card for the CoS value of the multicast video traffic enables networks to efficiently manage multicast video bandwidth demands on a network shared with VoIP and other Ethernet services.

Note Multicast priority queuing traffic uses port-based load-balancing over RPR and EtherChannel. Default multicast traffic is load-balanced over RPR, but not over EtherChannel.

Note Multicast priority queuing bandwidth should not be oversubscribed for sustained periods with traffic from multiple sources. This can result in reduced multicast priority queuing throughput.

Default Multicast QoS

Default multicast traffic is any multicast traffic (including flooded traffic) that is not classified as multicast priority queuing. The default multicast class also includes broadcast data traffic, control traffic, L2 protocol tunneling, and flooding traffic of the unknown MAC during MAC learning.

With no QoS configured (no multicast priority queuing and no output policy map) on the ML-Series card, the default multicast bandwidth is a 10 percent minimum of the total bandwidth.

When bandwidth is allocated to multicast priority queuing but no output policy map is applied, the default multicast congestion bandwidth is a minimum of 10 percent of the bandwidth that is not allocated to multicast priority queuing.

When an output policy-map is applied to an interface, default multicast and default unicast share the minimum bandwidth assigned to the default class. This default class is also known as the match-any class. The minimum bandwidth of default multicast is 10 percent of the total default class bandwidth.

Multicast Priority Queuing QoS Restrictions

The following restrictions apply to multicast priority queuing QoS:

- The bandwidth allocation and utilization configured for multicast priority queuing traffic is global and applies to all the ports on the ML-Series card, both POS and Fast Ethernet, regardless of whether these ports carry multicast priority queuing traffic. The rate of traffic can be reduced for all ports on the ML-Series card when this feature is configured. Default multicast traffic uses bandwidth only on the ports where it egresses, not globally like multicast priority queuing.
- Multicast priority queuing QoS is supported only for Layer 2 bridging.
- The ML-Series card supports a maximum of two multicast priority queuing classes.
- Unlike the rest of the ML-Series card QoS, multicast priority queuing QoS is not part of the Cisco IOS MQC.
- Priority-mcast bandwidth allocation is per port.

Configuring Multicast Priority Queuing QoS

To configure a priority class for multicast traffic, use the global configuration **cos priority-mcast** command defined in Table 12-5.

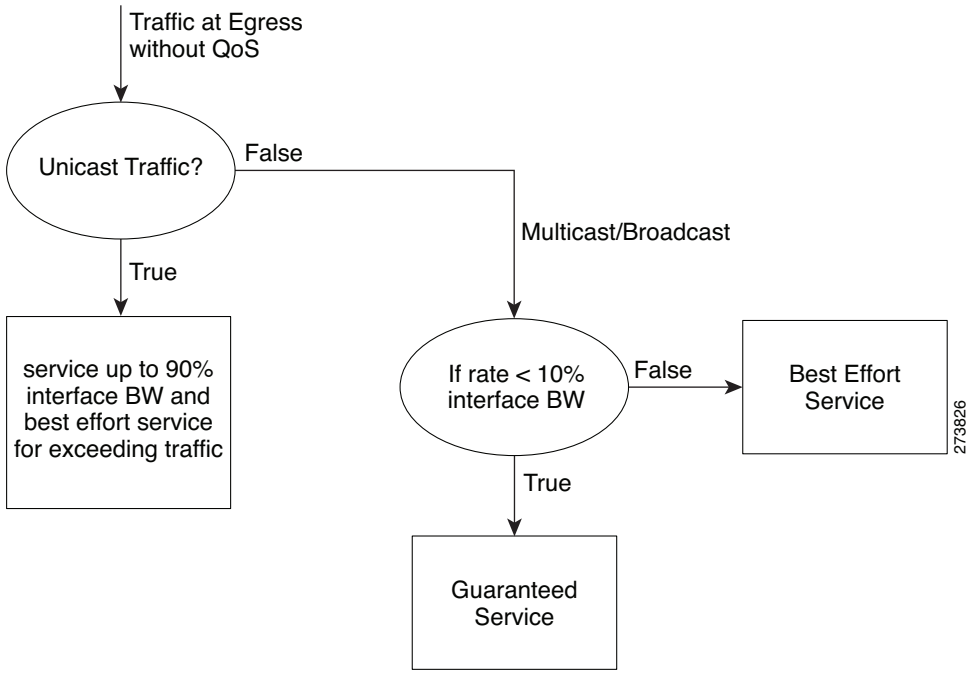
Table 12-5 CoS Multicast Priority Queuing Command

Command	Purpose
`Router (config)# [no] cos priority-mcast cos-value {bandwidth-kbps | mbps bandwidth-mbps | percent percent}`	*Creates a priority class of multicast traffic based on a multicast CoS value and* specifies a minimum bandwidth guarantee to a traffic class in periods of congestion. *cos-value* specifies the CoS value of multicast packets which will be given the bandwidth allocation. Matches only a single CoS of traffic (not a range). Supported CoS range is 0 to 7. A minimum bandwidth guarantee can be specified in kbps, in mbps, or by a percentage of the overall available bandwidth. Valid choices for the ML-Series card are: • Rate in kilobits per second • Rate in megabits per second • Percent of total available port bandwidth (1 to 100) Reentering the command with the same *cos-value* but a different bandwidth rate will modify the bandwidth of the existing class. Reentering the command with a different *cos-value* creates a separate multicast priority queuing class with a maximum of two multicast priority queuing classes. The **no** form of this command removes the multicast priority queuing class. **Note** The true configurable bandwidth in kilobits or megabits per second is per port and depends on how the ML-Series card is configured. The **show interface** command shows the maximum bandwidth of a port (for example, BW 100000 Kbit). The sum of all bandwidth and priority actions applied to the interface, plus the cos priority-mcast bandwidth, is not allowed to exceed the maximum bandwidth of the port. **Note** Attempting to configure a priority-mcast bandwidth that exceeds the true configurable bandwidth on any port will cause the priority-mcast configuration change to fail, and the multicast priority queuing bandwidth guarantee will not be changed.

QoS not Configured on Egress

The QoS bandwidth allocation of multicast and broadcast traffic is handled separately from unicast traffic. On each interface, the aggregate multicast and broadcast traffic are given a fixed bandwidth commit of 10% of the interface bandwidth. This is the optimum bandwidth that can be provided for traffic exceeding 10% of the interface bandwidth.

Figure 12-10 QoS not Configured on Egress



ML-Series Egress Bandwidth Example

This section explains with examples the utilization of bandwidth across different queues with or without Priority Multicast.

Case 1: QoS with Priority and Bandwidth Configured Without Priority Multicast

Strict Priority Queue is always serviced first. The remaining interface bandwidth is utilized to service other configured traffic.

In the following example, after servicing unicast `customer_voice` traffic, the remaining interface bandwidth is utilized for other WRR queues such as `customer_core_traffic`, `customer_data`, and `class-default` in the ratio of 1:3:5.

At any given time, the sum of the bandwidth assigned cannot exceed the interface bandwidth (in kbps). The bandwidth share allocated to `class-default` will be utilized by default unicast traffic (in this example, unicast traffic with CoS values other than 2, 5, 7) and all multicast/broadcast traffic (all CoS values). The default unicast and all multicast/broadcast traffic will be serviced in the ratio of 9:1.

For example, if 18x bandwidth is available after servicing priority unicast traffic (CoS 5), then the remaining bandwidth will be allocated as follows:

Unicast traffic with CoS 2 : 2x

Unicast traffic with CoS 7: 6x

Unicast default (without CoS 2, CoS 5, CoS 7): 9x

All multicast/broadcast (any CoS value): 1x

Example 12-14 QoS with Priority and Bandwidth Configured without Priority Multicast

```
!
class-map match-all customer_voice
  match cos  5
class-map match-all customer_data
  match cos  7
class-map match-all customer_core_traffic
  match cos  2
!
!
policy-map policy_egress_bandwidth
  class customer_core_traffic
    bandwidth   1000
  class customer_voice
    priority 1000
  class customer_data
    bandwidth   3000
  class class-default
    bandwidth   5000
!
!
interface POS0
 no ip address
  crc 32
 service-policy output policy_egress_bandwidth
!
```

Case 2: QoS with Priority and Bandwidth Configured with Priority Multicast

In this case, only multicast traffic of CoS 3 is allocated a guaranteed bandwidth. This multicast traffic will now participate in the queue along with other WRR queues. After servicing the `customer_voice` traffic, the remaining interface bandwidth is utilized for WRR queues, such as `customer_core_traffic`, `customer_data`, `class-default`, and multicast CoS 3 traffic in the ratio of 1:3:5:2.

At any given time, the sum of the bandwidth assigned cannot exceed the interface bandwidth (in kbps).

Example 12-15 QoS with Priority and Bandwidth configured with Priority Multicast

```
cos priority-mcast 3 2000
!
class-map match-all customer_voice
  match cos  5
class-map match-all customer_data
  match cos  7
class-map match-all customer_core_traffic
  match cos  2
!
!
```

```
policy-map policy_egress_bandwidth
  class customer_core_traffic
    bandwidth   1000
  class customer_voice
    priority 1000
  class customer_data
    bandwidth   3000
  class class-default
    bandwidth   5000
!
!
interface POS0
 no ip address
  crc 32
 service-policy output policy_egress_bandwidth
!
```

Understanding CoS-Based Packet Statistics

Note For IEEE 802.1Q (QinQ) enabled interfaces, CoS accounting is based only on the CoS value of the outer metro tag imposed by the service provider. The CoS value inside the packet sent by the customer network is not considered for CoS accounting.

Enhanced performance monitoring displays per-CoS packet statistics on the ML-Series card interfaces when CoS accounting is enabled. CoS-based traffic utilization is displayed at the Fast Ethernet interface or subinterface (VLAN) level, or at the POS interface level. It is not displayed at the POS subinterface level. RPR statistics are not available at the SPR interface level, but statistics are available for the individual POS ports that make up the SPR interface. EtherChannel (port-channel) and BVI statistics are available only at the member port level. Table 12-6 shows the types of statistics available at specific interfaces.

Table 12-6 Packet Statistics on ML-Series Card Interfaces

Statistics Collected	Fast Ethernet Interface	Fast Ethernet Subinterface (VLAN)	POS Interface	POS Subinterface
Input—Packets and Bytes	Yes	Yes	No	No
Output—Packets and Bytes	Yes	Yes	No	No
Drop Count—Packets and Bytes[1]	Yes	No	Yes	No

1. Drop counts only include discards caused by output congestion and are counted at the output interface.

CoS-based packet statistics are available through the Cisco IOS command-line interface (CLI) and Simple Network Management Protocol (SNMP), using an extension of the CISCO-PORT-QOS MIB. They are not available through Cisco Transport Controller (CTC).

Configuring CoS-Based Packet Statistics

Note For IEEE 802.1Q (QinQ) enabled interfaces, CoS accounting is based only on the CoS value of the outer metro tag imposed by the service provider. The CoS value inside the packet sent by the customer network is not considered for CoS accounting.

To enable CoS-based packet statistics on an interface, use the interface configuration level command defined in Table 12-7.

Table 12-7 CoS-Based Packet Statistics Command

Command	Purpose
`ML_Series(config-if)# cos accounting`	Enables CoS-based packet statistics to be recorded at the specific interface and for all the subinterfaces of that interface. This command is supported only in interface configuration mode and not subinterface configuration mode. The **no** form of the command disables the statistics.

After configuring CoS-based packet statistics on the ML-Series card, the statistics can be viewed through a variety of **show** commands. To display this information, use one of the commands in Table 12-8 in EXEC mode.

Table 12-8 Commands for CoS-Based Packet Statistics

Command	Purpose
`ML_Series# show interface type number cos`	Displays the CoS-based packet statistics available for an interface.
`ML_Series# show interface type number.subinterface-number cos`	Displays the CoS-based packet statistics available for a FastEthernet subinterface. POS subinterfaces are not eligible.

Example 12-16 shows examples of these commands.

Example 12-16 Commands for CoS-Based Packet Statistics Examples

```
ML_Series# show interface fastethernet 0.5 cos
FastEthernet0.5
    Stats by Internal-Cos
    Input: Packets         Bytes
        Cos 0: 31          2000
        Cos 1:
        Cos 2: 5           400
        Cos 3:
        Cos 4:
        Cos 5:
        Cos 6:
        Cos 7:
    Output: Packets        Bytes
        Cos 0: 1234567890  1234567890
        Cos 1: 31          2000
        Cos 2:
```

```
        Cos 3:
        Cos 4:
        Cos 5:
        Cos 6: 10          640
        Cos 7:

ML_Series# show interface fastethernet 0 cos
FastEthernet0
    Stats by Internal-Cos
    Input: Packets         Bytes
        Cos 0: 123         3564
        Cos 1:
        Cos 2: 3           211
        Cos 3:
        Cos 4:
        Cos 5:
        Cos 6:
        Cos 7:
    Output: Packets        Bytes
        Cos 0: 1234567890  1234567890
        Cos 1: 3           200
        Cos 2:
        Cos 3:
        Cos 4:
        Cos 5:
        Cos 6: 1           64
        Cos 7:
    Output: Drop-pkts      Drop-bytes
        Cos 0: 1234567890  1234567890
        Cos 1:
        Cos 2:
        Cos 3:
        Cos 4:
        Cos 5: 1               64
        Cos 6: 10              640
        Cos 7:

ML_Series# show interface pos0 cos
    POS0
    Stats by Internal-Cos
    Output: Drop-pkts          Drop-bytes
        Cos 0: 12              1234
        Cos 1: 31              2000
        Cos 2:
        Cos 3:
        Cos 4:
        Cos 5:
        Cos 6: 10              640
        Cos 7:
```

Understanding IP SLA

Cisco IP SLA, formerly known as the Cisco Service Assurance Agent, is a Cisco IOS feature to assure IP service levels. Using IP SLA, service provider customers can measure and provide service level agreements, and enterprise customers can verify service levels, verify outsourced service level agreements, and understand network performance for new or existing IP services and applications. IP SLAs use unique service level assurance metrics and methodology to provide highly accurate, precise service level assurance measurements.

Depending on the specific IP SLAs operation, statistics of delay, packet loss, jitter, packet sequence, connectivity, path, server response time, and download time are monitored within the Cisco device and stored in both CLI and SNMP MIBs. The packets have configurable IP and application layer options such as source and destination IP address, User Datagram Protocol (UDP)/TCP port numbers, a type of service (ToS) byte (including Differentiated Services Code Point [DSCP] and IP Prefix bits), Virtual Private Network (VPN) routing/forwarding instance (VRF), and URL web address.

IP SLAs uses generated traffic to measure network performance between two networking devices such as routers. IP SLAs starts when the IP SLAs device sends a generated packet to the destination device. After the destination device receives the packet, and depending on the type of IP SLAs operation, the device will respond with time-stamp information for the source to make the calculation on performance metrics. An IP SLAs operation is a network measurement to a destination in the network from the source device using a specific protocol such as UDP for the operation.

Because IP SLA is accessible using SNMP, it also can be used in performance monitoring applications for network management systems (NMSs) such as CiscoWorks2000 (CiscoWorks Blue) and the Internetwork Performance Monitor (IPM). IP SLA notifications also can be enabled through Systems Network Architecture (SNA) network management vector transport (NMVT) for applications such as NetView.

For general IP SLA information, refer to the Cisco IOS IP Service Level Agreements technology page at http://www.cisco.com/warp/public/732/Tech/nmp/ipsla. For information on configuring the Cisco IP SLA feature, see the "Network Monitoring Using Cisco Service Assurance Agent" chapter of the *Cisco IOS Configuration Fundamentals Configuration Guide, Release 12.2*. at: http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a008030c773.html.

IP SLA on the ML-Series

The ML-Series card has a complete IP SLA Cisco IOS subsystem and offers all the normal features and functions available in Cisco IOS Release 12.2S. It uses the standard IP SLA Cisco IOS CLI commands. The SNMP support will be equivalent to the support provided in the IP SLA subsystem 12.2(S), which is the rttMon MIB.

IP SLA Restrictions on the ML-Series

The ML-Series card supports only features in the Cisco IOS 12.2S branch. It does not support functions available in future Cisco IOS versions, such as the IP SLA accuracy feature or the enhanced Cisco IOS CLI support with updated IP SLA nomenclature.

Other restrictions are:

- Setting the CoS bits is supported, but set CoS bits are not honored when leaving or entering the CPU when the sender or responder is an ONS 15454, ONS 15454 SDH, ONS 15310-CL, or ONS 15310-MA platform. Set CoS bits are honored in intermediate ONS nodes.
- On RPR, the direction of the data flow for the IP SLA packet might differ from the direction of customer traffic.
- The system clock on the ML-Series card synchronizes with the clock on the TCC2/TCC2P card. Any NTP server synchronization is done with the TCC2/TCC2P card clock and not with the ML-Series card clock.

- The average Round Trip Time (RTT) measured on an ML-Series IP SLA feature is more than the actual data path latency. In the ML-Series cards, IP SLA is implemented in the software. The IP SLA messages are processed in the CPU of the ML-Series card. The latency time measured includes the network latency and CPU processing time. For very accurate IP SLA measurements, it is recommended that a Cisco Router or Switch be used as an external probe or responder to measure the RTT of the ML-Series cards in a network.