JUNIPER NETWORKS

ACCESS & AGGREGATION · EDGE · CORE · DATA CENTER · WAN · CAMPUS & BRANCH · CONSUMER & BUSINESS DEVICE

Junos® Fundamentals

DAY ONE: JUNOS QOS FOR IOS ENGINEERS

When it's time to swap out your Cisco routers with new Juniper Networks devices, use this book to optimize the QoS functionality of your new network.

By Venkatesh Krishnan

DAY ONE: JUNOS QoS FOR IOS ENGINEERS

Book #9 in the Junos Fundamentals series

QoS has become an integral part of network design given the volume of video and voice traffic over IP. While both IOS and Junos fundamentally support QoS across their platforms, there are still some differences in how the OSes are implemented and how they are understood.

Day One: Junos QoS for IOS Engineers addresses the needs of the IOS-trained engineer by providing side-by-side comparisons of QoS configurations and techniques in both IOS and Junos. In a few quick steps you can compare what you did yesterday with IOS to what you can do today with Junos. Along the way you'll find insights, tips, and no-nonsense explanations of what is taking place. If you are an engineer who is already familiar with IOS QoS, get ready to see the 'Junos way' in action, whether it's simply using a different syntax or a whole hierarachy.

> "This Day One book is perfect for the network engineer who would like to add Class-of-Service to their network, but only has familiarity with IOS. The side-by-side comparison between IOS and Junos CoS configurations makes it easy to understand and implement CoS on a Juniper device."
>
> Chris Jones, JNCIE-ENT #272,
> Author of *Day One: Junos for IOS Engineers*

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Design and deploy QoS on Junos devices within your network.
- Troubleshoot basic QoS issues with Junos troubleshooting procedures.
- Execute QoS specific Junos show commands.
- Create QoS configuration templates for deployment.

Juniper Networks Books are singularly focused on network productivity and efficiency. Peruse the complete library at www.juniper.net/books.

Published by Juniper Networks Books

ISBN 978-1936779543

51600

9 781936 779543

7100157

JUNIPER NETWORKS

Junos® Networking Technologies

Day One: Junos QoS for IOS Engineers

By Venkatesh Krishnan

JUNIPER NETWORKS

Published by Juniper Networks Books
Author: Venkatesh Krishnan
Technical Reviewers: Jack Parks
Editor in Chief: Patrick Ames
Copyeditor and Proofer: Nancy Koerbel
J-Net Community Manager: Julie Wider

About the Author
Venkatesh Krishnan is a Network Consultant with the Professional Services Organization at Juniper Networks specializing in design and implementation of enterprise and service provider networks. Prior to joining Juniper Networks, he has also worked in the Advanced Services team at Cisco Systems working as a Network Consulting Engineer in the Service Provider team. He has assisted in designing QoS solutions for customers both on Cisco and Juniper devices. He has been in the field of Networking and Security for over 10 years. He holds a CCIE-R&S #24243.

Author's Acknowledgments
There are a lot of people I would like to thank for their assistance. I would like to thank my family for their support and patience throughout this process, my manager, Craig Sirkin, for his encouragement and guidance and Patrick Ames and the *Day One* team for tirelessly working through this. Finally, I would like to thank my colleague, Jack Parks, whose input, tips, and advice in reviewing this information helped me convert it into a book.

ISBN: 978-1-936779-56-7 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-936779-57-4 (ebook)

Version History: v1 December 2012
2 3 4 5 6 7 8 9 10 #7100159-en

This book is available in a variety of formats at: http://www.juniper.net/dayone. Send your suggestions, comments, and critiques by email to dayone@juniper.net.

Welcome to Day One

This book is part of a growing library of *Day One* books, produced and published by Juniper Networks Books.

Day One books were conceived to help you get just the information that you need on day one. The series covers Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

The *Day One* library also includes a slightly larger and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a weeklong seminar.

You can obtain either series, in multiple formats:

- Download a free PDF edition at http://www.juniper.net/dayone.
- Get the ebook edition for iPhones and iPads from the iTunes Store. Search for Juniper Networks Books.
- Get the ebook edition for any device that runs the Kindle app (Android, Kindle, iPad, PC, or Mac) by opening your device's Kindle app and going to the Kindle Store. Search for Juniper Networks Books.
- Purchase the paper edition at either Vervante Corporation (www.vervante.com) or Amazon (www.amazon.com) for between $12-$28, depending on page length.
- Note that Nook, iPad, and various Android apps can also view PDF files.
- If your device or ebook app uses .epub files, but isn't an Apple product, open iTunes and download the .epub file from the iTunes Store. You can now drag and drop the file out of iTunes onto your desktop and sync with your .epub device.

What You Need to Know Before Reading This Book

Before reading this book, you should be familiar with the basic administrative functions of the Junos operating system, including the ability to work with operational commands and to read, understand, and change Junos configurations. There are several books in the *Day One* library on exploring and learning Junos, at www.juniper.net/dayone.

This book makes a few assumptions about you, the reader:

- You are familiar with implementing QoS on IOS devices.

- You know the basics of troubleshooting a QoS problem on an IOS device.
- You have a basic understanding of Junos.

After Reading This Book, You'll Be Able To:

- Design and deploy QoS on Junos devices within your network
- Troubleshoot basic QoS issues with Junos troubleshooting procedures
- Execute QoS specific Junos show commands
- Create QoS configuration templates for deployment

IOS and Junos

Given the volume of video and voice traffic over IP, QoS has become an integral part of network design. Most customer networks include a mix of Cisco and Juniper equipment, and while both Cisco and Juniper fundamentally support QoS across their platforms, there are still some differences in how they are implemented and understood that the reader will immediately see.

It can be easy to get lost in a lengthy IOS configuration file while trying to understand the specific QoS configuration attached to an interface, but Junos allocates a separate hierarchy to QoS. Junos also follows and implements a specific order when performing QoS.

In addition, there are some troubleshooting mechanisms that will provide faster resolution to a QoS problem in Junos, all of which are highlighted in this book.

This book does not delve deeply into the concepts of QoS or specific platform related troubleshooting. Instead, it guides you through building a QoS configuration in Junos based upon your IOS experience, and then provides a comparison of how the Junos implementation differs from IOS.

Chapter 1

Introduction

Imagine you are at home, watching high-def Netflix movies on your laptop while talking on the VoIP phone and browsing the Internet. Now, imagine everyone in your neighborhood doing the same thing, then everyone in your city, and so on. Let's impose the condition that it must be high quality high-def streaming video, clear voice quality on the VoIP phone, and no page cannot be displayed alerts while browsing. You've got it – when you start placing more and more demands on the Internet, it becomes essential to effectively identify each packet and treat it accordingly.

The ability to *identify specific packets and to process each packet based upon its identity* is the underlying principle of class of service or quality of service (QoS). Delay, jitter, packet loss, bandwidth requirements, and latency are the inherent parameters that govern traffic quality. QoS is a tool that helps control and manipulate the flow of traffic to achieve better performance.

MORE? See *Day One: Deploying Basic QoS* for a succinct explanation of QoS at http://www.juniper.net/dayone.

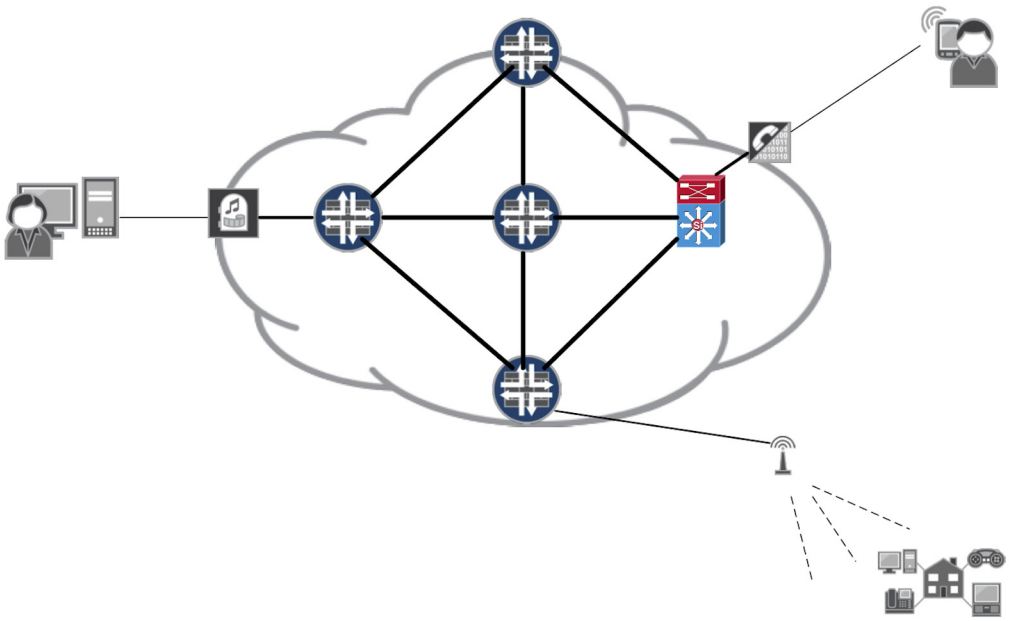

Figure 1.1 Internet Use

A basic QoS function can be broken into these five fundamental steps, which are used throughout the rest of this *Day One* book:

- Classification – This is where you try to identify the type of packet. The criteria for identifying the packet can range from source and destination IP addresses, DSCP, MPLS EXP bits to layer-4 ports, protocols, etc. *Classification* provides the ability to understand the different types of traffic patterns within your network. By clearly defining and classifying the traffic types you can exercise better control over how they are handled through your network.
- Queuing – Once the packets have been classified you can then define steps on how to process them. The first step is allocating the classes to the hardware queues on an interface (the number of hardware queues will depend upon the specific platform available).
- Scheduling and Prioritizing – Once the different classes have been assigned to defined queues, the next step is to define how each queue will be serviced and how many packets you need to take from each queue when sending them over the wire through the interface. *Prioritizing* also defines which queue gets serviced first and in what order they will be serviced.
- Bandwidth assignment –Bandwidth will be assigned according to the scheduling mechanism defined for each queue.
- Marking – Though not always mandatory, you can choose to acknowledge and pass the traffic with the same marking that it arrived with, or choose to rewrite the marking of the packet, or configure a marking on the packet if it has no marking. *Marking* will modify the DSCP/802.1Q/IP Precedence or MPLS EXP bit value. Often it is preferable to define specific marking for packets to enable the following network devices to honor and process the packet accordingly.

Most modern networks contain a medley of different vendor equipment including Cisco and Juniper. For IOS engineers the command line for Junos might seem different from IOS, but you can still configure Junos devices to perform the same functions. It can be a challenge sometimes to transition from one command line to another, (albeit an enjoyable challenge for engineers, nonetheless). This book attempts to ease that transition by providing a comparative study of how a QoS configuration is built both in IOS and in Junos.

It shows the IOS configurations first and then follows with the comparable Junos version, which includes tips and highlights for the IOS engineer.

MORE? *Day One: Junos for IOS Engineers* provides a good reference for cultivating an understanding of the nuances of Junos. In addition, *Day One: Configuring Junos Basics* can help create a launching pad to build a Junos configuration for any device. These and other *Day One* books on Junos can make switching from your IOS background to Junos much easier. See the complete *Day One* library at http://www.juniper.net/dayone.

Chapter 2

Class of Service Functionality in IOS and Junos

This chapter compares class of service functionality in the two OSes: first IOS, and then, in comparison, Junos. To get you started, we construct a simple QoS configuration and then allow you to build on your understanding by expanding the scope of the discussion in later chapters. For most practical purposes, Quality of Service (QoS) in IOS translates to Class-of-Service (CoS) in Junos, and those terms are used interchangeably in this book.

NOTE CoS in Cisco terminology indicates manipulating bits in Layer-2 only. Junos, on the other hand, simplifies the task and addresses all changes under Class-of-Service. This includes layer-2 and layer-3, along with MPLS EXP bits.

Figure 2.1 is a generic representation of what elements QoS will address in any network.

Figure 2.1 QoS in a Network

Class of Service Functionality in Cisco IOS

When you focus on MQC (Modular QoS CLI) there are three basic steps to configure and deploy QoS on Cisco devices. Let's follow the sequence.

Defining and Configuring Class-map

Class-maps are used to classify packets with the criterion for classification controlled by the match statements. The match command can match on numerous fields in the packet header, ranging from DSCP, MPLS EXP bits, source/destination ip addresses, and source/destination mac-addresses, to input interfaces and protocols.

MORE? The following URL gives a complete list of match statements http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfm-cli2.html. Be sure to check for the latest updates on the Cisco website.

Class-maps have a *match-all* or a *match-any* option, with the default being match-all. The match-all keyword is used when all of the conditions must be met, while match-any is used when only one out the many conditions can be met in order for a packet to be placed in

that traffic class.

The *match not* option is used to specify a match criterion that prevents a packet from being classified as a member of the class.

Let's review some samples in the next six examples.

Example 1

Example 1 illustrates how an access-list can be used to create a class-map for IP addresses:

```
Router(config)# class-map match-any voice
Router(config-cmap)# match protocol ip
Router(config-cmap)# match access-group 101

Router(config)# access-list 101 permit 10.10.10.0 0.0.0.255
Router(config)# access-list 101 permit 10.10.20.0 0.0.0.255
Router(config)# access-list 101 permit 172.16.31.0 0.0.0.255
```

Example 2

Example 2 illustrates how a combination of ip addresses and mac-addresses can be used:

```
Router(config)# class-map match-any class2
Router(config-cmap)# match protocol ip
Router(config-cmap)# match access-group 2
Router(config-cmap)# exit

Router(config)# class-map match-all class1
Router(config-cmap)# match class-map class2
Router(config-cmap)# match destination-address mac 00.00.00.00.00.00
Router(config-cmap)# exit

Router(config)# access-list 2 permit 10.10.0.0 0.0.255.255
Router(config)# access-list 2 permit 192.168.0.0 0.0.0.255
```

Example 3

Example 3 highlights the use of the "match not" option:

```
Router(config)# class-map noip
Router(config-cmap)# match not protocol ip
Router(config-cmap)# exit
```

Example 4

Examples 4 and 5 illustrate the use of ip precedence, dscp, and rtp ports in a class-map while Example 6 highlights the use of nested class-maps:

```
Router(config)# class-map match-all ex-4
Router(config-cmap)# match ip rtp 16384 16383
```

```
Router(config-cmap)# match precedence 5
Router(config-cmap)# exit
```

Example 5

```
Router(config)# class-map match-any ex-5
Router(config-cmap)# match dscp 0
Router(config-cmap)# match dscp 1
Router(config-cmap)# exit
```

Example 6

```
Router(config)# class-map match-any class-2
Router(config-cmap)# match ip rtp 16384 16383
Router(config-cmap)# match precedence 5
Router(config-cmap)# exit

Router(config)# class-map match-any class-1
Router(config-cmap)# match class-map class-2
Router(config-cmap)# match cos 5
Router(config-cmap)# exit
```

Defining and Configuring policy-map

Each policy-map will call several class-maps, and for each class-map bandwidth, buffer size and priority can be defined. It's a good practice to define the last class as "default" and assign it a certain bandwidth allocating all the traffic that has not been classified to this class. It is possible to configure nested policies, depending upon code and platform support. Packet marking is also configured at this stage.

The priority command sets the guaranteed bandwidth for the traffic. This option can be used with traffic that is sensitive to latency, like voice or video. The bandwidth command reserves bandwidth for that class of traffic while the queue limit specifies the maximum number of packets the queue can hold for that class of traffic. When bandwidth is specified as a percentage, then the available bandwidth is calculated based upon the interface bandwidth or as a percentage of the "bandwidth" configured under the interface. The following examples highlight some of the ways in which policy-maps can be created.

NOTE Hierarchical scheduling is available in Junos, but is beyond the scope of this book. See *The Juniper MX Series* by Douglas Hanks and Harry Reynolds, published in 2012 by O'Reilly Media for an entire chapter on H-CoS. http://www.juniper.net/books.

Example 1

Example 1 shows how the dscp values can be set for different classes of traffic:

```
Router# show run

policy-map voip
class voip-rtp
set dscp ef
class class-default
set dscp default
```

Example 2

Example 2 illustrates how the various options can be defined for each class including bandwidth and queue-limit:

```
Router(config)# policy-map policy1
Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 1000
Router(config-pmap-c)# queue-limit 20
Router(config-pmap)# exit

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

Example 3

Example 3 illustrates a nested policy along with the use of the priority option:

```
Router(config)# policy-map child
Router(config-pmap)# class voice
Router(config-pmap-c)# priority 50

Router(config)# policy-map parent
Router(config-pmap)# class class-default
Router(config-pmap-c)# service-policy child
```

Configuring the Service Policy

Applying the service policy to the interface is the final step once the packets have been classified and each class of traffic is allocated the right bandwidth along with what they will be marked as. In the event that a "bandwidth" command is specified under the interface, the value configured here is used to calculate the actual bandwidth for each class, in case bandwidth-percent is used. The service policy can be applied in either the input or output direction.

```
Router(config)# interface g1/0/0
Router(config-if)# service-policy output policy1
Router(config-if)# exit
```

This example illustrates how a service policy can be deployed.

Queuing and Scheduling

How packets are classified and marked has already been discussed here, but an explicit description of how the packets will be serviced on their way out has not. How will the packets that have been marked be moved out, and how will sensitive packets, like voice and video, be serviced differently when compared to other packets?

After the packets are processed they are stored in memory until the outgoing interface is available to transmit them. IOS creates a queue in order to manage the packets that are waiting to exit the interface. The basic queuing scheme uses FIFO (first in, first out) scheduling and services the packets in that order. This method cannot account for delay, jitter, and latency, and packet loss is a concern, given the finite storage available to hold the packets. Modifying the length of the queue will minimize packet loss but it comes at the cost of tampering with the latency and delay, hence the need to have multiple queues and a mechanism to determine how to service which queues at any given moment. IOS offers various queuing and scheduling techniques that determine how the packets exit the interface as summarized in Table 2.1.

Table 2.1 IOS Queing and Scheduling Characteristics

Queuing Method	Max. number of queues	Classification capabilities	How they are serviced
FIFO	1	Default	First packet to arrive is serviced first.
Priority Queuing (PQ)	4	Access-list	Always serves higher priority queues over lower priority queues.
Custom Queuing (CQ)	16	Access-list	Serves a configured number of bytes per queue, per round robin pass through the queues.
Weighted Fair Queuing (WFQ)	4096	Automatic (based on flows which are identified by src/dst address, port numbers, and protocol).	Queues with low volume and higher IP precedence get higher service compared to high volume but low precedence, with each flow using a different queue.
Class-Based Weighted Fair Queuing (CBWFQ)	64	Access-list/Class maps	Certain percentage of bandwidth is allocated to each queue.

Low Latency Queuing (LLQ)	N/A	Same as CBWFQ	Similar to CBWFQ, except some queues configured as "priority" are always getting served before other queues. It polices traffic with a configured priority bandwidth so as to not starve other queues.
Modified Deficit Round Robin (MDRR)	8	IP Precedence	Similar to CQ, but if any queue is serviced in excess of its configured value then less data is dequeued the next time the queue is serviced.

BEST PRACTICE

Changing the queue depth is not a vendor-specific issue – network traffic is affected regardless of what device is used. It is therefore essential to *study* the network traffic pattern and then assign a queue depth accordingly.

FIFO and WFQ are the default queuing methods on some router interfaces. LLQ, MDRR, and PQ are the only techniques that provide a high priority queue for delay sensitive traffic, while CBWFQ and LLQ are typically the only techniques configured via MQC. Not all queuing methods are available with every Cisco platform.

Example 1: FIFO

```
interface serial0/0
ip address 192.168.10.1 255.255.255.0
no fair-queue
hold-queue 20 out
```

Example 2: LLQ

```
access-list 101 permit udp any any range 16384 32767
access-list 102 permit tcp any any eq 23

class-map data
   match access-group 102

class-map voice
   match access-group 101

policy-map llq
  class voice
     priority percent 50
  class data
     bandwidth percent 25
  class class-default
    fair-queue

interface se0/0
 ip address 172.16.18.77 255.255.255.0
 service-policy output llq
```

The queues described here are created by the software in the router. When the scheduler or the scheduling algorithm decides which packet needs to be sent next, it is not directly transmitted out of the interface. Instead, the packet is moved from the interface software queue to the transmit queue, which is a small FIFO queue on each interface. This transmit queue is referred to as the *hardware queue*, *TX queue*, or *TX ring*. The primary purpose of the hardware queue is to drive the link utilization to 100 percent when packets are waiting to exit an interface.

NOTE Having multiple queuing techniques does not automatically translate to greater flexibility. This is because every queuing method is not available on every platform or in every version of code. Multiple queuing techniques are only an indication of how queuing methods have evolved over a period of time. Junos uses queue priority and WRR to service the queues with Strict-High getting the highest privilege. Strict-High, if configured, is similar to LLQ.

IOS Summary

Configuring QoS in IOS involves four broad steps:

- First, identify the queuing method.
- Second, define and configure a class-map.
- Third, define and configure a policy-map.
- Fourth, apply the service policy to the interface.

Class-maps can use different techniques to classify packets including access-lists, protocols, dscp values, etc. As most of you know, it is important to choose the correct queuing method to get optimum results.

Class of Service Functionality in Junos

First, Junos has a structured command line. If you notice the configurations of any device or in these pages, you will see the built in hierarchy. There are sections for interfaces, protocols, system configuration, chassis, etc. and each of these sections has numerous sub-sections. Quality of Service comes under the main section, *Class-of-Service*. The majority of the QoS configuration that needs to be defined and configured is done under this section. You will see that there are a few exceptions, but ultimately they tie back to Class-of-Service.

MORE? See the resources at the end of this book to learn more about Junos.

This section builds on the basic steps set forth in Chapter 1 that are needed to define and configure QoS. Each section elaborates a necessary step, and then, finally, builds the configuration to classify and mark the traffic.

Let's begin these steps to configure class of service on Junos.

Define and Configure Forwarding Classes

Forwarding class is the naming convention followed for each traffic pattern that we want to define and classify. The main function of a forwarding class is to determineto which packet the que will be assigned. Forwarding classes can also be considered as configuration components that represent the queues.

By default there are four forwarding classes defined. Multiple forwarding classes can be defined and assigned to the same queue. It is important to note that the number of forwarding classes does not need to equal the number of queues. The maximum number of queues varies depending upon the platform.

NOTE Since the number of queues and the maximum number of forwarding classes that are supported varies by platform, it will help to check the device specifications to see how many are supported.

```
user@R1> show class-of-service forwarding-class
Forwarding class                  ID      Queue  Restricted queue  Fabric priority
  best-effort                      0          0                 0             low
  expedited-forwarding             1          1                 1             low
  assured-forwarding               2          2                 2             low
  network-control                  3          3                 3             low
```

Configuring Forwarding Classes

In addition to assigning the forwarding class to the queue, the PLP (Packet Loss Priority) also needs to be defined. PLP defines the packet's drop eligibility. While this configuration does not automatically drop the packet, it provides the user with a mechanism to control what packets will be dropped in the event of congestion. Packets with a higher PLP will be dropped first during congestion in the network.

```
[edit]
user@R1#
class-of-service {
   forwarding-classes {
      class NC6 queue-num 3 priority high;
      class NC7 queue-num 3 priority high;
```

```
    class EF queue-num 1 priority high;
    class AF3 queue-num 2 priority low;
    class AF4 queue-num 2 priority low;
    class AF2 queue-num 2 priority low;
    class BE queue-num 0 priority low;
    class SCV queue-num 4 priority low;
}
```

Define and Configure Classification of Packets

There are three ways in which classification can be achieved:

- Fixed Classification – this is the simplest way to classify ingress packets because a single forwarding class can be assigned to an interface or VLAN. This method is useful when all the inbound traffic from a neighbor needs to be assigned to a specific queue.
- Behavior Aggregate (BA) Classification – when traffic from a neighbor already has CoS markings, you can use the markings to classify packets. The CoS markings that can be matched on include: DSCP, 802.1p, MPLS EXP, IP Precedence, and the 802.1ad drop eligible indicator bit. This method requires less packet analysis and is more efficient in high traffic volumes, especially in the network core.
- Multi-field (MF) Classification – MF classifier offers the most granular control when classifying packets. It can match on one or more fields in the packet header. Source/ Destination IP address, destination port number, protocol are examples of how MF classification can be done. MF classification is achieved by defining firewall filters and applying the filter to the interface or VLAN.

If an interface has a BA classifier and a MF classifier configured, then the BA classification is performed first. But if both classifiers match traffic then the MF classifier will override the BA classifier.

NOTE Another item to be cautious about is the number of TCAM entries that will be consumed when multiple ports are defined under the MF classifier and applying the filter to the interface. Depending upon the platform, this could trigger inconsistent behavior, as most low end devices are limited in TCAM space.

Configuring Fixed Classifier

```
[edit]
user@R1#
class-of-service {
```

```
    interfaces {
        ge-0/0/0 {
            unit 0 {
                forwarding-class assured-forwarding;
            }
        }
    }
}
```

Configuring BA Classifier

When creating a custom classifier, it is useful to use the related default classifier of the same CoS type, because you only have to change a few settings. The following snippet lists the code points and what forwarding class is associated with the default dscp classifier, as well as the PLP defined by default.

```
user@R1> show class-of-service classifier
Classifier: dscp-default, Code point type: dscp, Index: 7
  Code point         Forwarding class                  Loss priority
  000000             best-effort                       low
  000001             best-effort                       low
  000010             best-effort                       low
  000011             best-effort                       low
  000100             best-effort                       low
  000101             best-effort                       low
  000110             best-effort                       low
  000111             best-effort                       low
  001000             best-effort                       low
  001001             best-effort                       low
  001010             assured-forwarding                low
  001011             best-effort                       low
  001100             assured-forwarding                high
  001101             best-effort                       low
  001110             assured-forwarding                high
  001111             best-effort                       low
  010000             best-effort                       low
  010001             best-effort                       low
  010010             best-effort                       low
  010011             best-effort                       low
  010100             best-effort                       low
  010101             best-effort                       low
  010110             best-effort                       low
  010111             best-effort                       low
  011000             best-effort                       low
  011001             best-effort                       low
  011010             best-effort                       low
  011011             best-effort                       low
  011100             best-effort                       low
  011101             best-effort                       low
  011110             best-effort                       low
  011111             best-effort                       low
```

```
  100000              best-effort                    low
  100001              best-effort                    low
  100010              best-effort                    low
  100011              best-effort                    low
  100100              best-effort                    low
  100101              best-effort                    low
  100110              best-effort                    low
  100111              best-effort                    low
  101000              best-effort                    low
  101001              best-effort                    low
  101010              best-effort                    low
  101011              best-effort                    low
  101100              best-effort                    low
  101101              best-effort                    low
  101110              expedited-forwarding           low
  101111              best-effort                    low
  110000              network-control                low
  110001              best-effort                    low
  110010              best-effort                    low
  110011              best-effort                    low
  110100              best-effort                    low
  110101              best-effort                    low
  110110              best-effort                    low
  110111              best-effort                    low
  111000              network-control                low
  111001              best-effort                    low
  111010              best-effort                    low
  111011              best-effort                    low
  111100              best-effort                    low
  111101              best-effort                    low
  111110              best-effort                    low
  111111              best-effort                    low
```

The **show class-of-service classifier** command also provides a list for 802.1p, ip precedence, and EXP (but the output is not shown here).

So applying this information allows the user to modify only a few terms and combine them with the default classifier to get a greater scope for classification. Here the "import default" section is configured:

```
[edit]
user@R1#
class-of-service {
   classifiers {
      dscp VOICE-BA-CLASSIFIER {
         import default;
         forwarding-class expedited-forwarding {
            loss-priority low code-points [ cs5 ef ];
         }
         forwarding-class assured-forwarding {
            loss-priority low code-points [ cs1 af11 af12 af13 cs2 af21 af22 af23 ];
            loss-priority high code-points [ cs3 af31 af32 af33 cs4 af41 af42 af43 ];
         }
```

```
            forwarding-class best-effort {
                loss-priority low code-points be;
            }
            forwarding-class network-control {
                loss-priority low code-points cs7;
                loss-priority high code-points cs6;
            }
        }
    }
    interfaces {
        ge-0/0/0 {
            unit 0 {
                classifiers {
                    dscp VOICE-BA-CLASSIFIER;
                }
            }
        }
}
```

Configuring MF Classifier

The firewall filter can be configured and applied to a Layer 3 interface (with an IP address configured), or to a Layer 2 interface (with no IP address configured):

```
[edit]
user@R1#
firewall {
    family ethernet-switching {
        filter VOICE-MF-CLASSIFIER-L2 {
            term ACCEPT-VOICE {
                from {
                    protocol udp;
                    destination-port 10000-65000;
                }
                then {
                    accept;
                    forwarding-class expedited-forwarding;
                    loss-priority low;
                }
            }
            term ACCEPT-VOICE-CONTROL {
                from {
                    destination-port [ 5060 5070-5074 5400-5800 32768-65535 ];
                }
                then {
                    accept;
                    forwarding-class assured-forwarding;
                    loss-priority low;
                }
            }
            term ACCEPT-ALL-BE {
                then {
                    accept;
```

```
                forwarding-class best-effort;
                loss-priority low;
             }
          }
       }
    }
}

interfaces {
    ge-0/0/0 {
       unit 0 {
          family ethernet-switching {
            filter {
               input VOICE-MF-CLASSIFIER-L2;
             }
             }
          }
     }
}
```

Define and Configure Scheduling

Once the packets have been classified and assigned to the various queues, the next step entails:

- which queue will be serviced first (define priority)
- how long the queues will be serviced for (define bandwidth)
- how other queues will be serviced (Weighted Round Robin)

In order to achieve these, you must first configure a scheduler, then associate the scheduler with a scheduler-map, and finally, apply the scheduler-map to the interface.

Configuring Scheduler

Specifying the bandwidth and buffer-size is defined under the scheduler. *Transmit rate* is the amount of bandwidth allocated to a queue while the *buffer-size* refers to the amount of data that can be stored during congestion. *Weighted Round Robin* (WRR) is the algorithm that defines how each queue is serviced. While the more important queues are serviced first, the algorithm ensures that lower priority queues are not starved for bandwidth. This ensures that a balance is maintained while transporting the packets through the device.

Queue priority indicates the relative importance of the queue when compared to other queues. Each queue is serviced depending upon the priority assigned to it. There are five supported priorities, from the highest to the lowest priorities:

- Strict-high
- High
- Medium-high
- Medium-low
- Low

Strict-High has the highest priority and provides for low latency queuing to be processed first, so it is ideal for voice traffic. But since Strict-High is a special setting, it has the capability to consume the bandwidth of the entire interface, and should be used cautiously. Strict-High and High share an underlying hardware queue that results in them sharing precedence to transmit packets.

NOTE Queue priority and packet loss priority (PLP) are different. Queue priority dictates the relative importance any queue gets with respect to other queues, while PLP refers to the probability of any packet being dropped in the event of any congestion. PLP is used as part of congestion control.

NOTE The obvious question that comes to mind is: How do you define congestion? Congestion can be defined and configured by the operator for every network, and drop profiles in Chapter 5 will give you a better understanding of how this is achieved.

There are four packet loss priorities that can be configured:

- Low
- Medium-low
- Medium-high
- High

Configuring the buffer-size and/or the transmit-rate, along with the priority, are the key steps in determining which queue is serviced at any point of time. It also helps in establishing how long a specific queue will be serviced before moving to the next queue preventing bandwidth starvation. The example here illustrates how schedulers can be defined and configured:

```
[edit]
user@R1#
class-of-service {
   schedulers {
      EF-SCHED {
         buffer-size percent 20;
         priority strict-high;
```

```
        }
        VC-SCHED {
            transmit-rate remainder;
            buffer-size remainder;
            priority low;
        }
        NC-SCHED {
            buffer-size percent 10;
            priority high;
        }
        AF-SCHED {
            transmit-rate percent 50;
            buffer-size percent 50;
            priority low;
        }
    }
}
```

Configuring Scheduler-map

Schedulers are designed to manage traffic for a particular forwarding class. Scheduler maps provide the link between the schedulers and the forwarding class.

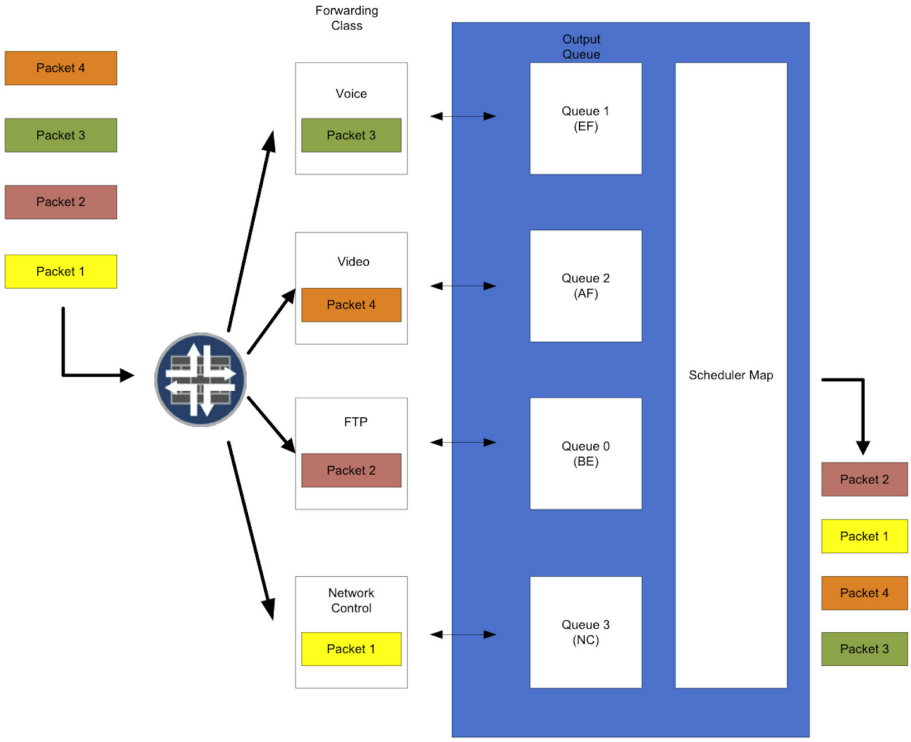

Figure 2.2 Forwarding Class, Queuing, and Scheduling

Figure 2.2 depicts how packets arriving at the router are mapped to their respective forwarding classes based upon their pre-defined classification criteria. Each forwarding class is associated with a queue. The scheduler-map associated with the outgoing interface then transmits the packets accordingly, based upon the criteria defined under each scheduler.The configuration here illustrates how the scheduler-map is associated with the schedulers:

```
[edit]
user@R1#
class-of-service {
    scheduler-maps {
      VOICE-SCHED-MAP {
         forwarding-class best-effort scheduler VC-SCHED;
         forwarding-class network-control scheduler NC-SCHED;
         forwarding-class expedited-forwarding scheduler EF-SCHED;
         forwarding-class assured-forwarding scheduler AF-SCHED;
      }
   }
}
```

Deploying Scheduler-map

The final step is to apply the scheduler-map to the interface. Scheduler maps are implicitly applied in the outbound direction on the specific interface because scheduling occurs at the output of an interface. Here's an example:

```
[edit]
user@R1#
class-of-service {
   interfaces {
      ge-0/0/0 {
         scheduler-map VOICE-SCHED-MAP;
         }
   }
}
```

Define and Configure Rewrite Rules

The purpose of packet rewrite, or *marking,* is to efficiently convey the packet's CoS profile to the next-hop router to provide a consistent end-to-end CoS policy for packets traversing the network. DSCP, MPLS EXP, IP Precedence, 802.1p, or the 802.1ad drop eligibility indicator bit, are options in packet markings. The import default serves the same function as under the BA classifier. This snippet shows the default dscp:

```
user@R1> show class-of-service rewrite-rule
Rewrite rule: dscp-default, Code point type: dscp, Index: 31
  Forwarding class                  Loss priority       Code point
  best-effort                       low                 000000
```

```
  best-effort                    high            000000
  expedited-forwarding           low             101110
  expedited-forwarding           high            101110
  assured-forwarding             low             001010
  assured-forwarding             high            001100
  network-control                low             110000
  network-control                high            111000
```

Configuring Rewrite Rule

The example here illustrates how the rewrite rules can be configured. It shows how packets assigned to each forwarding-class will be marked in combination with its PLP value:

```
[edit]
user@R1#
class-of-service {
   rewrite-rules {
      dscp VOICE-DSCP-REWRITE {
         import default;
         forwarding-class expedited-forwarding {
            loss-priority low code-point ef;
         }
         forwarding-class assured-forwarding {
            loss-priority low code-point cs3;
            loss-priority high code-point cs3;
         }
         forwarding-class network-control {
            loss-priority low code-point cs7;
            loss-priority high code-point cs6;
         }
         forwarding-class best-effort {
            loss-priority low code-point be;
         }
      }
   }
   interfaces {
      ge-0/0/0 {
         unit 0  {
            rewrite-rules {
               dscp VOICE-DSCP-REWRITE;
            }
        }
    }
}
```

Summary

There are four main steps involved in defining and configuring QoS on any Junos device:

- Defining and configuring forwarding classes
- Defining and configuring schedulers
- Defining and configuring scheduler maps
- Defining and configuring rewrite rules

The way that QoS is processed in Junos is illustrated in Figure 2.3.

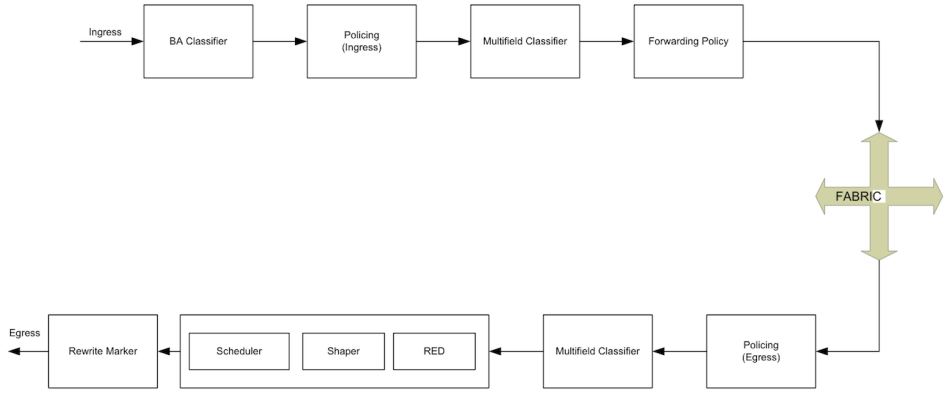

Figure 2.3 QoS Processing in Junos

To process QoS in Junos, it's important to first lay out the requirements, then to discuss and decide on the forwarding classes, define the schedulers before allocating bandwidth to them, and finally to associate the scheduler maps with the forwarding classes and define how you want to rewrite the packets.

Once you have a plan laid out it's just a matter of translating that plan onto the device and configuring it. This methodology provides a workable structure around the rather amorphous concept of QoS.

For the IOS engineer, working in Junos highlights these apparent differences:

- As most configuration is localized to the Class-of-Service hierarchy in Junos, you do not need to scroll through the whole configuration to locate specific access-lists or another class-map. Although this might not aid you while you're configuring QoS, you will appreciate Junos when you try to put together and understand the QoS deployed on an interface.

- If a configuration is not supported, then that configuration will not be accepted when you build it. This provides a certain degree of freedom while configuring with preemptive troubleshooting.
- The indentation and structured outlook enhances readability of any configuration because you can see the heirarchical structure of Junos.
- The ability to monitor real-time traffic on interfaces makes it easier to identify and troubleshoot any issues you may have.

Chapter 3

Comparison of IOS and Junos with Basic Tuple

Chapter 1 listed the five basic steps that are essential to designing QoS in any network. Now let's use those steps and see how they fit within IOS and Junos.

In IOS

Classification is achieved in the first step by defining class-maps. Queuing, scheduling, and prioritizing are achieved in the next step by defining the queuing algorithm, while bandwidth allocation and marking are achieved in the final step along with attaching the policy to the interface.

Classification	Configure class-maps
Queuing Scheduling & Prioritizing	For most purposes Class Based Weighted Fair Queuing (CBWFQ) or Low Latency Queuing (LLQ) is used. The specific queuing method also highlights how scheduling is done.
Bandwidth Assignment Marking	Configure policy-map For each class of traffic define bandwidth For each class of traffic define what the packet marking should be Apply the policy-map to the interface

In Junos

Classification and Queuing are achieved in the first step while defining forwarding classes and using BA Classifier/MF Classifier. Scheduling, prioritizing, and Bandwidth Assignment are achieved in the next step while defining the schedulers and the scheduler-map and assigning it to the interface. Finally, marking is achieved by defining the rewrite rules and applying them to the interface.

Classification Queuing	Configure forwarding class under class-of-service Configure BA classifier and apply it to the interface under class-of-service or configure MF classifier firewall filter and apply it to the physical interface
Scheduling & Prioritizing Bandwidth Assignment	Configure schedulers under class-of-service Configure scheduler-map under class-of-service Assign a scheduler-map to the interface under class-of-service
Marking	Configure rewrite rule under class-of-service and apply it to the interface under class-of-service

Translating IOS to Junos

Now that we understand how the Junos and Cisco CLI relate to the basic tuple, the next question that arises is how to translate the Cisco configuration to Junos. The table below summarizes how this is done.

	Cisco IOS	Junos
Classification	Option under "class-map" – access-list, protocols, etc.	The first step, which is a preparation defining the forwarding classes. The next step is classification, which can be achieved either by BA classifier under Class-of-service | classifiers or MF classifier under Firewall.
Queuing and Scheduling	Depending upon the platform and code FIFO, LLQ, CBWFQ, CQ, etc. are available.	Defining a scheduler under Class-of-service | schedulers and then a scheduler-map under Class-of-service | scheduler-map.
Prioritizing	LLQ is specified using "priority" bandwidth, while the priority of other queues varies depending upon the algorithm like FIFO, CBWFQ, WFQ etc.	The queue priority can be specified under Class-of-service | schedulers.
Bandwidth Assignment	Defining the bandwidth or bandwidth percent under policy-map.	Defining a scheduler under Class-of-service | schedulers and then a scheduler-map under Class-of-service | scheduler-map and applying it to the interface under Class-of-service | interfaces.
Marking	"Set" command under policy-map.	Option under Class-of-service | rewrite.
Where is it applied	Defining a service policy under the interface in inbound or outbound direction.	Option under Class-of-service | interfaces for BA classifier or under Interfaces for MF Classifier.

Summary

Junos provides a structured and more granular way to define and configure QoS. By being able to elaborately specify the scheduling, queuing, and prioritizing techniques used, the user can exercise greater control over how they would like the packets to be handled. Functionally Junos will be able to achieve the same configuration of any Cisco IOS device.

Chapter 4

Policing and Shaping

Traffic conditioning is an important step in managing congestion in the network. Policing and shaping are two methods that will help reduce congestion by continuously measuring the rate at which data is sent or received. Policing discards excess packets while not imposing any delay to the conforming traffic, thus enforcing service level agreements. Shaping, on the other hand, buffers excess traffic and releases it based on the shaping rate.

Policer

Policing applies a hard limit to the rate at which traffic arrives or leaves an interface. Packets are either dropped (hard policing) or re-classified (soft policing) if they do not conform to the constraints. Policing is helpful under certain conditions where the neighboring network could send more traffic than the contract specification. Policing will then enforce the contract, thus protecting the network from being overrun with too much traffic. Policer can be applied in an inbound or outbound direction.

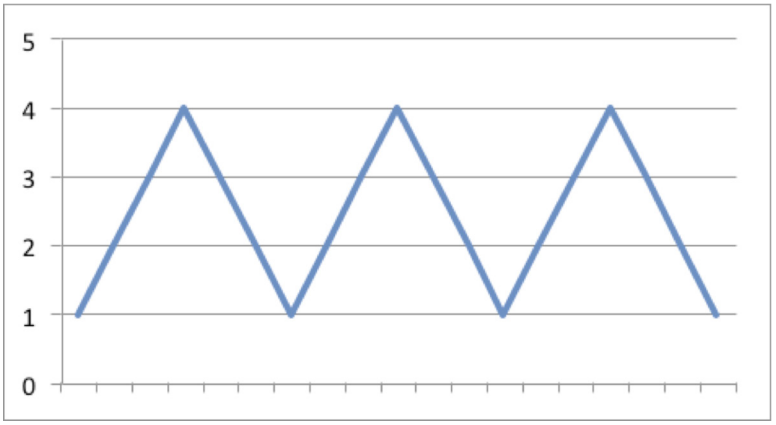

Figure 4.1 Before Traffic Policing

An example of traffic policing would be a customer who had purchased services for 4 Mbps from the carrier, and the customer's contract with the carrier says they will not exceed traffic beyond 4 Mbps. In the event traffic exceeds 4 Mbps, then the carrier will not honor the traffic and will cap it to remain at 4 Mbps. This will prevent the carrier's network from being overloaded with excess traffic.

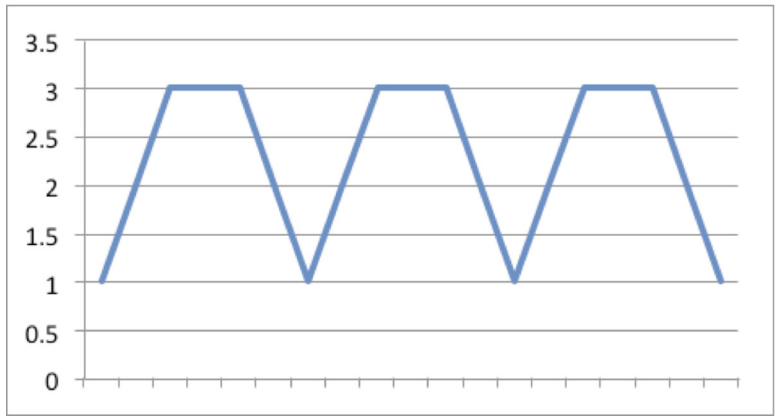

Figure 4.2 After Traffic Policing - Out-of-Profile

In IOS

Policing on Cisco IOS can be achieved in two ways:

- Committed Access Rate (CAR)
- Class Based Policing

NOTE Please refer to the platform type and specific version of IOS code for support of CAR.

Example 1 illustrates the use of Class based policing, while Example 2 highlights the use of CAR.

Example 1: Class Based Policing

```
Router(config)# policy-map police
Router(config-pmap)# class c2
Router(config-pmap-c)# police 8000 2000 4000 conform-action transmit exceed-action
set-qos-transmit 4 violate-action drop
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface fastethernet 0/0
Router(config-if)# service-policy input police
```

Example 2: CAR

```
Router# show run

interface Hssi0/0/0
 rate-limit input 15000000 2812500 2812500 conform-action transmit exceed-action drop
 ip address 192.168.24.1 255.255.255.252
 rate-limit output 15000000 2812500 2812500 conform-action transmit exceed-action drop
```

In Junos

There are three modes in which policing can operate in Junos:

1. Single rate two color – consisting of one rate threshold (CIR) and one burst size (CBS)

2. Single rate tricolor marking – consisting of one rate threshold (CIR) and two burst sizes (CBS & PBS)

3. Two rate tricolor marking – consisting of two rate thresholds (CIR & PIR) and two burst sizes (CBS & PBS)

Policing can either be conducted:

- Directly on an interface, or
- Configured through a firewall filter and then applied to an interface.

For tricolor marking, Junos has two modes of operation:

- Color-blind – where the policer does not consider any previous marking of packets and rewrites the packet's marking with a PLP value based on the policer's settings.
- Color-aware- where the policer considers any existing coloring on a packet and the PLP values will be based on the combination of the previous value and the result of passing through the given policer.

In addition, there are two possible variations you can use when policing using a firewall filter:

- Aggregate policer for two color marking – when multiple terms are defined under the firewall and each term calls upon the policer, then the default behavior is to associate the bandwidth-limit and burst size for every specific term. But if you need to summarize and have all terms comply to the same policer and be associated with total bandwidth-limit, then specify the "filter-specific" keyword under the policer.
- Aggregate policer for two color or tricolor marking – when there are multiple protocol families and multiple units under an interface, and each of them has the firewall filter with the policer configured, then it is preferable to aggregate them together in order to have the same bandwidth constraint imposed on the physical interface rather than at the logical (per unit or per protocol family) level. This is achieved by specifying the "physical-interface-policer" keyword under the policer.

A two-color policer being applied directly to the interface can also be

aggregated when applied to multiple protocol families under the same interface. This is achieved by specifying the keyword "logical-interface-policer" under the policer.

Interface policers and firewall filter policers can both exist under the interface at the same time. They will be processed in the following order:

Inbound	Outbound
Interface Policer	Firewall filter
Firewall Filter	Interface Policer

Configuring a Policer

The examples below illustrate the different methods for configuring a policer – using a firewall filter, applying it directly on an interface, or applying single rate tri-color marking. Example 1 demonstrates how the policer is applied directly to the interface. It specifies that if the traffic exceeds the defined rate then all traffic will be dropped. Example 2 illustrates how a firewall filter is applied to an interface while restricting the amount of DNS and ICMP (router advertisement and redirect) traffic to that interface. Example 3 illustrates the use of a single rate tri-color marking.

Example 1: Directly on an Interface

```
 [edit]
user@R1#
firewall {
   policer my-2color-policer {
      if-exceeding {
         bandwidth-limit 1m;
         burst-size-limit 2k;
      }
      then discard;
   }
}

interfaces {
   ge-1/0/0 {
      unit 0 {
         family inet {
            policer {
               input my-2color-policer;
            }
            address 172.22.54.2/24;
         }
      }
   }
}
```

Example 2: Within a Firewall Filter

```
 [edit]
user@R1#
firewall {
   policer RATE-LIMIT-PROTECT {
      filter-specific;
      if-exceeding {
         bandwidth-limit 2m;
         burst-size-limit 500k;
      }
      then discard;
   }
   filter ROUTER-PROTECT {
      term DNS {
         from {
            protocol udp;
            source-port 53;
         }
         then {
            policer RATE-LIMIT-PROTECT;
            accept;
         }
      }
      term ICMP {
         from {
            protocol icmp;
            icmp-type-except [ router-advertisement redirect ];
         }
         then {
            policer RATE-LIMIT-PROTECT;
            accept;
         }
      }
}

interfaces {
   ge-1/0/0 {
      unit 0 {
         family inet {
            filter {
               input ROUTER-PROTECT;
            }
            address 172.22.54.1/24;
         }
      }
   }
}
```

Example 3: Single-rate Tricolor marking

```
 [edit]
user@R1#
firewall {
```

```
    three-color-policer my-srTCM-policer {
        single-rate {
            color-aware;
            committed-information-rate 1m;
            committed-burst-size 5k;
            excess-burst-size 10k;
        }
    }
    filter ROUTER-PROTECT {
        term A {
            then {
                three-color-policer {
                single-rate my-srTCM-policer;
            }
        }
    }
}

interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                filter {
                    input ROUTER-PROTECT;
                }
                address 172.22.54.3/24;
            }
        }
    }
}

class-of-service {
    tri-color;
}
```

Shaper

Shaping also defines a limit to the rate at which traffic can be transmitted, but unlike policing, it acts on traffic that has already been granted access to a queue and is awaiting access to transmission resources. Shaping is useful when the neighboring network is policing or is slower in accepting traffic. Under such conditions a shaper can delay traffic, thus preventing it from getting dropped. Therefore, shaping can be less aggressive than policing and can have fewer of the negative side effects. As shaping refers to buffering packets, it applies to outbound traffic only. It is important to note that since shaping delays packets from being transmitted in the event of congestion, it has a negative effect on real time traffic like voice, which is sensitive to delay.

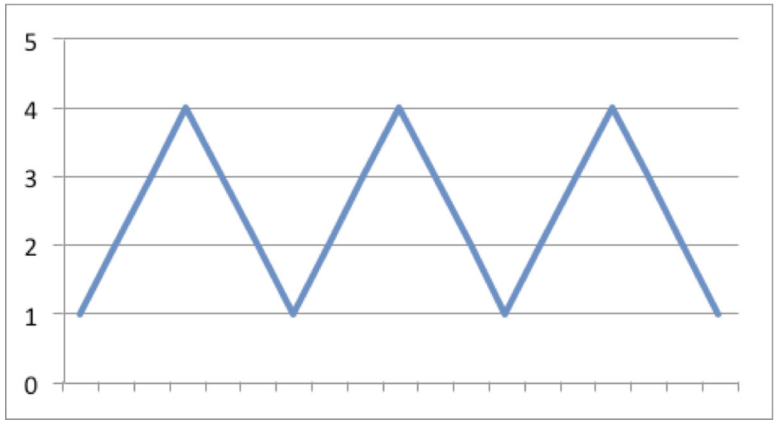

Figure 4.3 Traffic Pattern Before Shaping

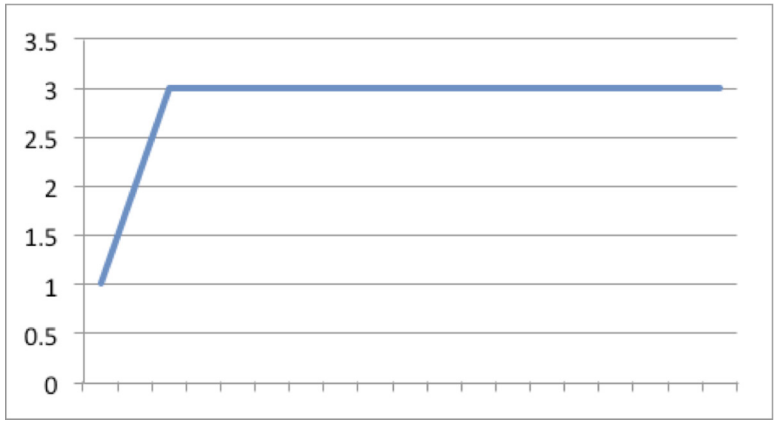

Figure 4.4 Traffic Pattern After Shaping

In IOS

Cisco IOS supports four different shaping tools, but the most widely used are Class Based Shaping and FRTS.

Shaping Tool	What Queue does it support
Generic Traffic Shaping (GTS)	WFQ
Class Based (CB) Traffic Shaping	FIFO, WFQ, CBWFQ, LLQ
Distributed Traffic Shaping (DTS)	FIFO, WFQ, CBWFQ, LLQ
Frame Relay Traffic Shaping (FRTS)	FIFO, WFQ, CBWFQ, LLQ, PQ, CQ

NOTE Please refer to the platform specifications on which shaping tools are supported for each platform.

Example 1 illustrates the use of class-based shaping while Example 2 demonstrates the use of generic traffic shaping.

Example 1: CB Traffic Shaping

```
Router(config)# policy-map shape
Router(config-pmap)# class test
Router(config-pmap-c)# shape average 384000 15440
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface Serial 0/0
Router(config-if)# service output shape
```

Example 2: GTS

```
Router# show run

access-list 101 permit udp any any
interface FastEthernet0/0
 traffic-shape group 101 1000000 125000 125000
!
interface FastEthernet0/1
 traffic-shape rate 5000000 625000 625000
```

In Junos

Shaping can be configured in the following ways:

- By defining traffic-control-profiles under the class-of-service or under the `class-of-service` schedulers.
- Under the `class-of-service| interfaces` for any specific interfaces.

The shaper permits traffic to be transmitted until the configured PIR (shaping rate). But once the PIR is exceeded then the packets are not transmitted until the traffic rate falls below the PIR. They are stored in a buffer. Out of the three options, defining shaping under traffic control profiles not only gives you more control but also has the additional option of specifying the guaranteed rate (CIR).

Example 1 illustrates the use of traffic-control-profiles, Example 2 `class-of-service|schedulers`, and Example 3 `class-of-service| interfaces`.

Example 1

```
[edit]
user@R1#
class-of-service {
   traffic-control-profiles {
      my-L1-profile {
         shaping-rate 500m;
      }
      my-L2-profile {
         shaping-rate 200m;
         guaranteed-rate 100m;
      }
```

```
        my-L3-profile {
            scheduler-map my-sched-map;
            shaping-rate 80m;
            guaranteed-rate 50m;
        }
    }
    interfaces {
        interface-set my-interface-set {
            output-traffic-control-profile my-L2-profile;
        }
        ge-1/0/4 {
            output-traffic-control-profile my-L1-profile;
            unit 0 {
                output-traffic-control-profile my-L3-profile;
            }
            unit 1 {
                output-traffic-control-profile my-L3-profile;
            }
        }
    }
}
```

Example 2

```
[edit]
user@R1# show class-of-service
schedulers {
    test {
        shaping-rate 32k;
    }
}
```

Example 3

```
[edit]
user@R1# show class-of-service
interfaces {
    ge-0/0/1 {
        shaping-rate 16k;
    }
}
```

Summary

Traffic shaping and policing are available techniques to help manage congestion in any network. Each technique is useful, has its own advantages and disadvantages, and has been deployed in customer environments.

	Policing	Shaping
Ingress	Can be configured	As it stores the packet before forwarding, it is not applicable in Ingress mode.
Egress	Can be configured	Can be configured

Chapter 5

Congestion Avoidance

The possibility of a large amount of traffic arriving at a router will likely lead to congestion. In the event of congestion, the queues will fill up quickly, and, with no buffer available, the packets will begin to get dropped. This mechanism, referred to as tail drop, offers no control over which packets will be dropped. Random Early Detection (RED) is a mechanism that helps determine actions at the onset of congestion by selectively dropping packets as the buffers fill.

In IOS

Weighted Random Early Detection (WRED) on a Cisco IOS device is achieved by defining a minimum threshold, a maximum threshold, and configuring the drop probability denominator. When congestion is below the minimum threshold no packets are dropped, while all packets are dropped when congestion is more than the maximum threshold defined. But when congestion is between the minimum and maximum thresholds defined, packets are dropped at the rate of one every drop probability denominator number of packets. Packets can be treated differently by using a different WRED profile for each IP Precedence/DSCP value. The examples below illustrate a few scenarios for deploying WRED.

Example 1

```
Router# show run

interface se0/0
 ip address 10.200.14.250 255.255.255.252
 random-detect
 random-detect precedence 0 32 256 100
 random-detect precedence 1 64 256 100
 random-detect precedence 2 96 256 100
 random-detect precedence 3 120 256 100
```

Example 2

```
Router# show run

interface se0/0
 ip address 10.200.13.250 255.255.255.252
 random-detect
 random-detect dscp-based
 random-detect dscp af21 24 40
 random-detect dscp af23 24 40
```

Example 3

```
Router# show run
```

```
policy-map Express
class Enterprise
bandwidth percent 25
random-detect dscp-based
random-detect dscp af21 24 40 10
random-detect dscp af22 28 40 10
random-detect dscp af23 30 40 10

class Business
bandwidth percent 30
random-detect dscp-based
random-detect dscp af11 22 38 10
random-detect dscp af12 24 38 10
random-detect dscp af13 28 38 10

class Data
bandwidth percent 30
random-detect dscp-based
random-detect dscp af31 26 34 10
random-detect dscp af32 28 34 10
random-detect dscp af33 30 34 10

class default
fair-queue
random-detect dscp-based
```

Example 4

```
Router# show run

interface Serial1/0
 random-detect
 random-detect flow
 random-detect flow average-depth-factor 8
 random-detect flow count 16
```

MORE? Flow-based WRED is also available. Flow-based WRED is a feature that forces WRED to afford greater fairness to all flows on an interface in regard to how packets are dropped. For more detail, please refer to http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconav_ps1835_TSD_Products_Configuration_Guide_Chapter.html#wp1005869 .

In Junos

Drop profiles perform RED/WRED by specifying the parameters for dropping traffic when congestion occurs. Multiple drop profiles can be configured and then applied to each queue.

There are two ways in which drop profiles can be configured:

- Segmented – In this method the drop probability percentage increases in steps when the buffer fills to the configured level.
- Interpolated – In this method the drop probability percentage increases proportionately between configured values as the buffer fills to the configured level.

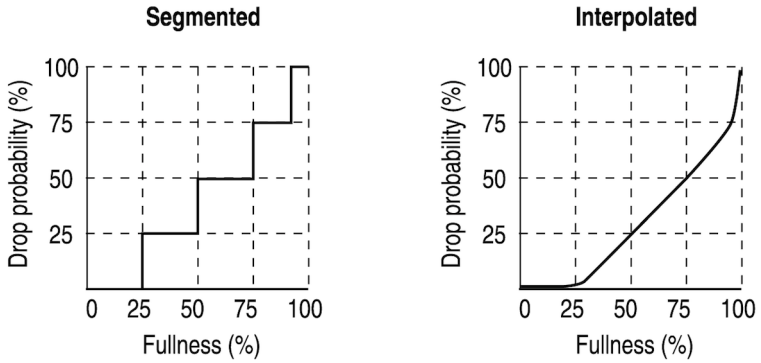

Figure 5.1 Drop Profiles

Drop profiles are created and associated with schedulers. A single scheduler can be configured to have multiple drop profiles for different traffic flows. The flexibility of specifying the packet loss priority (PLP) and the protocol is available while associating the drop profiles with the schedulers.

Configuring Segmented Drop Profile

```
[edit]
user@R1#
class-of-service {
   drop-profiles {
      HIGH {
            fill-level 25 drop-probability 25;
            fill-level 50 drop-probability 50;
            fill-level 75 drop-probability 75;
            fill-level 95 drop-probability 100;
      }
      LOW {
            fill-level 70 drop-probability 100;
      }
   }
   schedulers {
      AF-SCHED {
         transmit-rate percent 20;
         excess-rate percent 35;
         buffer-size temporal 15k;
         priority low;
         excess-priority high;
```

```
            drop-profile-map loss-priority high protocol any drop-profile HIGH;
        }
        BE-SCHED {
            transmit-rate remainder;
            excess-rate percent 30;
            buffer-size remainder;
            priority low;
            excess-priority low;
            drop-profile-map loss-priority low protocol any drop-profile LOW;
        }
        GENERAL-SCHED {
            priority low;
            drop-profile-map loss-priority any protocol any drop-profile LOW;
        }
    }
}
```

Configuring Interpolated Drop Profile

```
[edit]
user@R1#
class-of-service {
    drop-profiles {
        HIGH {
            interpolate {
                fill-level [ 0 10 15 ];
                drop-probability [ 0 90 100 ];
            }
        }
        MED-HIGH {
            interpolate {
                fill-level [ 15 20 25 ];
                drop-probability [ 0 90 100 ];
            }
        }
        MED-LOW {
            interpolate {
                fill-level [ 50 70 ];
                drop-probability [ 1 100 ];
            }
        }
        LOW {
            interpolate {
                fill-level [ 70 100 ];
                drop-probability [ 1 100 ];
            }
        }
    }
    schedulers {
        AF-SCHED {
            transmit-rate percent 20;
            excess-rate percent 35;
            buffer-size temporal 15k;
            priority low;
```

```
            excess-priority high;
            drop-profile-map loss-priority high protocol any drop-profile HIGH;
            drop-profile-map loss-priority medium-high protocol any drop-profile MED-HIGH;
            drop-profile-map loss-priority medium-low protocol any drop-profile MED-LOW;
            drop-profile-map loss-priority low protocol any drop-profile LOW;
        }
        BE-SCHED {
            transmit-rate remainder;
            excess-rate percent 30;
            buffer-size remainder;
            priority low;
            excess-priority low;
            drop-profile-map loss-priority high protocol any drop-profile HIGH;
            drop-profile-map loss-priority medium-high protocol any drop-profile MED-HIGH;
            drop-profile-map loss-priority medium-low protocol any drop-profile MED-LOW;
            drop-profile-map loss-priority low protocol any drop-profile LOW;
        }
        GENERAL-SCHED {
            priority low;
            drop-profile-map loss-priority any protocol any drop-profile LOW;
        }
    }
}
```

Summary

Congestion in any network is a reality that cannot be denied. While adding bandwidth is one option, it is always the more expensive one. Once there is congestion in any network then packets will be dropped. Congestion avoidance illustrates techniques which will allow us greater control over which specific packets can be dropped. The user can configure the network devices to drop higher loss priority packets when the network hits a certain threshold. Random Early Detection (RED) and/or Weighted Random Early Detection (WRED) are the techniques used to achieve this.

Congestion avoidance can be configured by configuring random detect for specific user traffic under the interfaces on IOS, while on Junos it can be achieved by specifying:

- Segmented Drop Profile
- Interpolated Drop Profile

Each method has its own advantages.

Chapter 6

Troubleshooting

If you run into some problems with class-of-service then how do you find out what went wrong? When a problem has been detected then here are some basic steps to follow in order to narrow down the potential cause of the issue.

IOS Troubleshooting Commands

Most IOS engineers follow this general procedure to troubleshoot QoS issues. Troubleshooting commands for IOS are listed in Table 6.1.

1. Identify the primary interfaces that are involved in carrying the traffic that has been affected.

2. Check the interface configuration to see if there is any service policy applied to the interface, and check to see if they are applied in the inbound or outbound direction.

3. Check the policy-map and the appropriate class-map configurations.

4. Check the interface counters to see if packet counts are increasing. This can be done by looking at the counters.

5. Check the policy-map interface command to see which queue is being serviced after the policy map has been attached to the interface.

6. Check the interface queue counters to see if the packets are going into the correct queue on egress and are in alignment with what is to be expected. This will help determine if packet classification or rewriting is an issue.

7. If the packet counters are incrementing as expected, and the packets are going to the appropriate queue but they are still causing performance issues, then it could be a hardware problem. Trace the path of the packet to determine any internal ASIC that could lead to this issue. At this stage involve Cisco TAC to resolve.

Table 6.1 IOS Troubleshooting Commands

Cisco IOS Commands	How will it help
Show run interface <interface-type>	This command will help to identify what service policy has been applied to the interface and whether it is inbound or outbound.
Show policy-map interface <interface-type>	This command will provide information about the various classes of traffic under this interface, along with the packets that are matched and packets that are marked, shaped, or policed. This command will also give information regarding the queuing method for every class.
Show interface<interface-type>	This command will provide information about packets sent/received along with the queuing type on the interface (FIFO etc.) along with whether WRED is enabled.

Show queue “serial 0/0”	This command highlights the queuing method along with information on buffer drops, input and output queue, availa bandwidth, etc.
Clear interface counters	This command will clear the interface statistics for all the counters on all the interfaces.
Show controllers “serial 0/0”	This command will list the size of the TX ring. The “tx_limi keyword will tell how many packets are being held by the T ring.
Show queuing interface “serial 0/0”	This command gives information about packets that are wa in a queue on the interface, queuing strategy, the tail drops p dscp value, and WRED information.

Junos Troubleshooting Commands

A similar procedure can be adopted to troubleshoot QoS related issues on Junos (most of the configuration appears under the class-of-service section).

1. Identify what are the primary interfaces that are involved in carrying the traffic that has been affected.

2. Check the interface configuration to see if there are any firewall filters, MF Classifiers, or policers applied to the interface.

3. Check the interface under the class-of-service configuration to see if there are any BA classifiers, rewrite-rules, or scheduler-maps configured.

4. Check the interface counters to see if packet counts are increasing in both directions. This can be done by either looking at the counters repeatedly or by monitoring the interface traffic in real time.

5. Check the interface queue counters to see if the packets are going into the correct queue on egress and if it is in alignment with what is to be expected. This will help determine if packet classification or rewriting is an issue.

6. In case there is a firewall filter or MF Classifier applied to the interface, then check firewall logs to see if any counters have incremented. If packet counting has been enabled then the firewall logs will reveal which counter is being incremented.

7. If all this does not help then sometimes the issue could be on the forwarding table, PFE, or the DPC. At this stage involve JTAC to resolve the issue.

Instead of listing the show commands in a table, we’ll review the show commands that are helpful in Junos troubleshooting so you can isolate problems pertaining to class of service/quality of service. Each of the

necessary show commands are detailed in the rest of this chapter:

- monitor interface traffic
- show interfaces <interface-type> extensive
- show interfaces <interface-type> extensive | find "queue counters"
- show class-of-service forwarding class
- show configuration class-of-service
- show interfaces queue <interface-type>
- clear interfaces statistics all
- show firewall
- show configuration interface <interface-type>
- show configuration class-of-service interface

monitor interface traffic

This command will help you monitor the input and output packets going through any interface on a device running Junos. The command can be monitored for real time traffic as the counters increase as packets go through the interface:

```
user@R1> monitor interface traffic
Bytes=b, Clear=c, Delta=d, Packets=p, Quit=q or ESC, Rate=r, Up=^U, Down=^D
R1                          Seconds: 27                 Time: 19:11:19
Interface    Link  Input packets        (pps)     Output packets        (pps)
 lc-0/0/0      Up              0                             0
 pfh-0/0/0     Up              0                             0
 xe-0/0/0    Down              0          (0)                0          (0)
 xe-0/0/1    Down              0          (0)                0          (0)
 xe-0/0/2    Down              0          (0)                0          (0)
 xe-0/0/3    Down              0          (0)                0          (0)
 ge-1/0/0      Up       35761707         (11)         35312140         (11)
 ge-1/0/1      Up          77976          (0)           737191          (0)
 ge-1/0/2      Up              8          (0)           442317          (0)
 ge-1/0/3    Down              0          (0)                0          (0)
 ge-1/0/4    Down              0          (0)                0          (0)
 ge-1/0/5    Down              0          (0)                0          (0)
 ge-1/0/6      Up         238472          (0)           238440          (0)
 ge-1/0/7    Down              0          (0)                0          (0)
 ge-1/0/8      Up         238440          (0)           238472          (0)
 ge-1/0/9    Down              0          (0)                0          (0)
 ge-1/0/10     Up              0          (0)                0          (0)
 ge-1/0/11   Down              0          (0)                0          (0)
 ge-1/1/0      Up              0          (0)                0          (0)
 ge-1/1/1    Down              0          (0)                0          (0)
 ge-1/1/2    Down              0          (0)                0          (0)
```

```
 ge-1/1/3     Down              0          (0)              0          (0)
 ge-1/1/4     Down              0          (0)              0          (0)
 ge-1/1/5     Down              0          (0)              0          (0)
 ge-1/1/6     Down              0          (0)              0          (0)
 ge-1/1/7     Down              0          (0)              0          (0)
 ge-1/1/8     Down              0          (0)              0          (0)
 ge-1/1/9     Down              0          (0)              0          (0)
 ge-1/1/10    Down              0          (0)              0          (0)
 ge-1/1/11    Down              0          (0)              0          (0)

Bytes=b, Clear=c, Delta=d, Packets=p, Quit=q or ESC, Rate=r, Up=^U, Down=^D
```

show interfaces <interface-type> extensive

This command is useful to find out the input/output packets:

```
user@R1> show interfaces ge-1/0/0 extensive
Physical interface: ge-1/0/0, Enabled, Physical link is Up
  Interface index: 144, SNMP ifIndex: 512, Generation: 147
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, BPDU Error: None, MAC-REWRITE
Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online, Speed-negotiation: Disabled, Auto-
MDIX: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x0
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 5c:5e:ab:00:cd:60, Hardware address: 5c:5e:ab:00:cd:60
  Last flapped   : 2012-09-05 19:47:21 UTC (1w4d 23:26 ago)
  Statistics last cleared: Never
  Traffic statistics:
   Input  bytes  :           1865678156                 4600 bps
   Output bytes  :           1849565880                 4392 bps
   Input  packets:             35763065                   11 pps
   Output packets:             35313479                   10 pps
   IPv6 transit statistics:
    Input  bytes  :                    0
    Output bytes  :                    0
    Input  packets:                    0
    Output packets:                    0
  Dropped traffic statistics due to STP State:
   Input  bytes  :                    0
   Output bytes  :                    0
   Input  packets:                    0
   Output packets:                    0
  Input errors:
    Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Policed discards: 0, L3
incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0, FIFO errors: 0, Resource
errors: 0
  Output errors:
   Carrier transitions: 3, Errors: 0, Drops: 0, Collisions: 0, Aged packets: 0, FIFO
```

```
errors: 0, HS link CRC errors: 0, MTU errors: 0, Resource errors: 0
  Egress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                2906                 2906                    0
    1 expedited-fo                  0                    0                    0
    2 assured-forw                  0                    0                    0
    3 network-cont           35310568             35310568                    0
  Queue number:         Mapped forwarding classes
    0                   best-effort
    1                   expedited-forwarding
    2                   assured-forwarding
    3                   network-control
  Active alarms  : None
  Active defects : None
  MAC statistics:                      Receive         Transmit
    Total octets                    2509372810       2477853994
    Total packets                     35763067         35313481
    Unicast packets                   35394125         34944569
    Broadcast packets                     1434             1470
    Multicast packets                   367508           367442
    CRC/Align errors                         0                0
    FIFO errors                              0                0
    MAC control frames                       0                0
    MAC pause frames                         0                0
    Oversized frames                         0
    Jabber frames                            0
    Fragment frames                          0
    VLAN tagged frames                       0
    Code violations                          0
  Filter statistics:
    Input packet count                35763055
    Input packet rejects                     0
    Input DA rejects                         0
    Input SA rejects                         0
    Output packet count                             35313470
    Output packet pad count                                0
    Output packet error count                              0
    CAM destination filters: 0, CAM source filters: 0
  Autonegotiation information:
   Negotiation status: Complete
   Link partner:
     Link mode: Full-duplex, Flow control: Symmetric/Asymmetric, Remote fault: OK
   Local resolution:
      Flow control: Symmetric, Remote fault: Link OK
  Packet Forwarding Engine configuration:
   Destination slot: 1
  CoS information:
    Direction : Output
    CoS transmit queue               Bandwidth              Buffer Priority   Limit
                              %            bps     %           usec
    0 best-effort            95      950000000    95              0      low    none
    3 network-control         5       50000000     5              0      low    none
  Interface transmit statistics: Disabled
```

```
  Logical interface ge-1/0/0.0 (Index 330) (SNMP ifIndex 561) (Generation 139)
    Flags: SNMP-Traps 0x0 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :           1865678156
     Output bytes  :           1847343804
     Input  packets:             35763065
     Output packets:             35313479
    Local statistics:
     Input  bytes  :             25260720
     Output bytes  :             30300888
     Input  packets:               370422
     Output packets:               370346
    Transit statistics:
     Input  bytes  :           1840417436                 4600 bps
     Output bytes  :           1817042916                 4392 bps
     Input  packets:             35392643                   11 pps
     Output packets:             34943133                   10 pps
    Protocol inet, MTU: 1500, Generation: 160, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Addresses, Flags: Is-Preferred Is-Primary
       Destination: 10.254.254.0/30, Local: 10.254.254.1, Broadcast: 10.254.254.3,
Generation: 142
   Protocol multiservice, MTU: Unlimited, Generation: 161, Route table: 0
     Flags: Is-Primary
     Policer: Input: __default_arp_policer__
```

show interfaces <interface-type> extensive | find "queue counters"

This command helps you find whether the packets are in the right queue in the egress:

```
user@R1> show interfaces ge-1/0/0 extensive | find "queue counters"
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                2906                 2906                    0
    1 expedited-fo                  0                    0                    0
    2 assured-forw                  0                    0                    0
    3 network-cont           35311240             35311240                    0
  Queue number:         Mapped forwarding classes
    0                   best-effort
    1                   expedited-forwarding
    2                   assured-forwarding
    3                   network-control
  Active alarms  : None
  Active defects : None
  MAC statistics:                      Receive         Transmit
    Total octets                    2509420312       2477900586
    Total packets                     35763744         35314145
    Unicast packets                   35394795         34945226
    Broadcast packets                     1434             1470
    Multicast packets                   367515           367449
    CRC/Align errors                         0                0
    FIFO errors                              0                0
    MAC control frames                       0                0
```

```
    MAC pause frames                        0                0
    Oversized frames                        0
    Jabber frames                           0
    Fragment frames                         0
    VLAN tagged frames                      0
    Code violations                         0
  Filter statistics:
    Input packet count                35763732
    Input packet rejects                    0
    Input DA rejects                        0
    Input SA rejects                        0
    Output packet count                              35314134
    Output packet pad count                                 0
    Output packet error count                               0
    CAM destination filters: 0, CAM source filters: 0
  Autonegotiation information:
    Negotiation status: Complete
    Link partner:
      Link mode: Full-duplex, Flow control: Symmetric/Asymmetric, Remote fault: OK
    Local resolution:
       Flow control: Symmetric, Remote fault: Link OK
  Packet Forwarding Engine configuration:
    Destination slot: 1
  CoS information:
    Direction : Output
    CoS transmit queue               Bandwidth              Buffer Priority   Limit
                              %            bps     %           usec
    0 best-effort            95      950000000    95              0      low    none
    3 network-control         5       50000000     5              0      low    none
  Interface transmit statistics: Disabled

  Logical interface ge-1/0/0.0 (Index 330) (SNMP ifIndex 561) (Generation 139)
    Flags: SNMP-Traps 0x0 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :           1865713264
     Output bytes  :           1847378386
                          Input packets:        35763738
```

show class-of-service forwarding class

This command provides information on all the forwarding classes available (default and user defined):

```
user@R1> show class-of-service forwarding-class
Forwarding class ID    Queue  Restricted queue  Fabric priority  Policing priority
  best-effort     0       0          0                low              normal
  expedited-
  forwarding      1       1          1                low              normal
  assured-
  forwarding      2       2          2                low              normal
  network-
```

```
  control          3       3          3                low                 normal
```

show configuration class-of-service –

This command shows the entire class of service configuration on the device. Individual sections can also be observed by specifying the keyword after class-of-service. For example, classifiers, rewrite-rules, etc.:

```
user@R1> show configuration class-of-service
classifiers {
    dscp VOICE-BA-CLASSIFIER {
        import default;
        forwarding-class VOICE-EF-CLASS {
            loss-priority low code-points [ cs5 ef ];
        }
        forwarding-class assured-forwarding {
            loss-priority low code-points [ cs1 af11 af12 af13 cs2 af21 af22 af23 ];
            loss-priority high code-points [ cs3 af31 af32 af33 cs4 af41 af42 af43 ];
        }
        forwarding-class best-effort {
            loss-priority low code-points be;
        }
        forwarding-class network-control {
            loss-priority low code-points cs7;
            loss-priority high code-points cs6;
        }
    }
}
forwarding-classes {
    queue 5 VOICE-EF-CLASS;
}
interfaces {
    ge-* {
        scheduler-map VOICE-SCHED-MAP;
        unit 0 {
            classifiers {
                dscp VOICE-BA-CLASSIFIER;
            }
            rewrite-rules {
                dscp VOICE-DSCP-REWRITE;
            }
        }
    }
}
rewrite-rules {
    dscp VOICE-DSCP-REWRITE {
        import default;
        forwarding-class VOICE-EF-CLASS {
            loss-priority low code-point ef;
        }
        forwarding-class assured-forwarding {
            loss-priority low code-point cs3;
```

```
            loss-priority high code-point cs3;
        }
        forwarding-class network-control {
            loss-priority low code-point cs7;
            loss-priority high code-point cs6;
        }
        forwarding-class best-effort {
            loss-priority low code-point be;
        }
    }
}
scheduler-maps {
    VOICE-SCHED-MAP {
        forwarding-class best-effort scheduler CONTROL;
        forwarding-class network-control scheduler NC;
        forwarding-class VOICE-EF-CLASS scheduler EF;
        forwarding-class assured-forwarding scheduler AF;
    }
}
schedulers {
    EF {
        buffer-size percent 20;
        priority strict-high;
    }
    CONTROL {
        transmit-rate remainder;
        buffer-size {
            remainder;
        }
        priority low;
    }
    NC {
        buffer-size percent 15;
        priority high;
    }
    AF {
        transmit-rate percent 40;
        buffer-size percent 40;
        priority low;
    }
                    }
```

show interfaces queue <interface-type>

This command details the specific queues associated with the interface – how many packets and bytes are queued and transmitted together with the RED dropped packets:

```
user@R1> show interfaces queue ge-1/0/0
Physical interface: ge-1/0/0, Enabled, Physical link is Up
```

```
  Interface index: 144, SNMP ifIndex: 512
Forwarding classes: 16 supported, 4 in use
Egress queues: 8 supported, 4 in use
Queue: 0, Forwarding classes: best-effort
  Queued:
    Packets              :                  2906                    0 pps
    Bytes                :                191844                    0 bps
  Transmitted:
    Packets              :                  2906                    0 pps
    Bytes                :                191844                    0 bps
    Tail-dropped packets :                     0                    0 pps
    RED-dropped packets  :                     0                    0 pps
     Low                 :                     0                    0 pps
     Medium-low          :                     0                    0 pps
     Medium-high         :                     0                    0 pps
     High                :                     0                    0 pps
    RED-dropped bytes    :                     0                    0 bps
     Low                 :                     0                    0 bps
     Medium-low          :                     0                    0 bps
     Medium-high         :                     0                    0 bps
     High                :                     0                    0 bps
Queue: 1, Forwarding classes: expedited-forwarding
  Queued:
    Packets              :                     0                    0 pps
    Bytes                :                     0                    0 bps
  Transmitted:
    Packets              :                     0                    0 pps
    Bytes                :                     0                    0 bps
    Tail-dropped packets :                     0                    0 pps
    RED-dropped packets  :                     0                    0 pps
     Low                 :                     0                    0 pps
     Medium-low          :                     0                    0 pps
     Medium-high         :                     0                    0 pps
     High                :                     0                    0 pps
    RED-dropped bytes    :                     0                    0 bps
     Low                 :                     0                    0 bps
     Medium-low          :                     0                    0 bps
     Medium-high         :                     0                    0 bps
     High                :                     0                    0 bps
Queue: 2, Forwarding classes: assured-forwarding
  Queued:
    Packets              :                     0                    0 pps
    Bytes                :                     0                    0 bps
  Transmitted:
    Packets              :                     0                    0 pps
    Bytes                :                     0                    0 bps
    Tail-dropped packets :                     0                    0 pps
    RED-dropped packets  :                     0                    0 pps
     Low                 :                     0                    0 pps
     Medium-low          :                     0                    0 pps
     Medium-high         :                     0                    0 pps
     High                :                     0                    0 pps
    RED-dropped bytes    :                     0                    0 bps
     Low                 :                     0                    0 bps
```

```
     Medium-low          :                    0                    0 bps
     Medium-high         :                    0                    0 bps
     High                :                    0                    0 bps
Queue: 3, Forwarding classes: network-control
  Queued:
    Packets              :             35312968                   11 pps
    Bytes                :           3184095280                 8216 bps
  Transmitted:
    Packets              :             35312968                   11 pps
    Bytes                :           3184095280                 8216 bps
    Tail-dropped packets :                    0                    0 pps
    RED-dropped packets  :                    0                    0 pps
     Low                 :                    0                    0 pps
     Medium-low          :                    0                    0 pps
     Medium-high         :                    0                    0 pps
     High                :                    0                    0 pps
    RED-dropped bytes    :                    0                    0 bps
     Low                 :                    0                    0 bps
     Medium-low          :                    0                    0 bps
     Medium-high         :                    0                    0 bps
     High                :                    0                    0 bps
```

show firewall

In Junos, if the firewall filter has terms with "count" turned on, then those counters will be indicated, and if the firewall includes policers with the count keyword then all the policer counters will be enabled and displayed under show firewall. Here are samples:

Firewall filter configuration example

```
filter CoS-count--ge-1/1/4_inbound {
      term 00 {
          from {
              dscp 000000;
          }
          then count dscp-00;
      }
      term 01 {
          from {
              dscp 000001;
          }
          then count dscp-01;
      }
      term 02 {
          from {
              dscp 000002;
          }
          then count dscp-02;
      }
      term 03 {
```

```
            from {
                dscp 000003;
            }
            then count dscp-03;
        }
        term 04 {
            from {
                dscp 000004;
            }
            then count dscp-04;
        }
        term 05 {
            from {
                dscp 000005;
            }
            then count dscp-05;
        }
        term else-accept {
            then accept;
        }
    }
```

Output

```
user@R1> show firewall

Filter: __default_bpdu_filter__

Filter: CoS-count--ge-1/1/4_inbound
Counters:
Name                                                Bytes              Packets
dscp-00                                             14000                  100
dscp-01                                                 0                    0
dscp-02                                                 0                    0
dscp-03                                                 0                    0
dscp-04                                                 0                    0
dscp-05                                                 0                    0

Filter: TCM_PLP-count--ge-1/0/4_outbound
Counters:
Name                                                Bytes              Packets
PLP-high                                                0                    0
PLP-low                                             15180                  114
PLP-med-high                                            0                    0

Filter: FC_PLP-count--ge-1/0/4_outbound
Counters:
Name                                                Bytes              Packets
BE-data-high                                            0                    0
BE-data-low                                             0                    0
Pri-data-high                                           0                    0
Pri-data-low                                            0                    0
Voice-high                                              0                    0
```

```
Voice-low                                              0                    0

Filter: my_MF_filter-ge-1/0/0.0-i
Counters:
Name                                                 Bytes              Packets
tcp-count-ge-1/0/0.0-i                                   0                    0
tcp80-count-ge-1/0/0.0-i                             14000                  100
udp-count-ge-1/0/0.0-i                                   0                    0
Policers:
Name                                               Packets
my-2color-policer-tcp-ge-1/0/0.0-i                       0
```

show configuration interface <interface-type>

This command shows what firewall filter, MF Classifier, or policer is configured under the specific interface:

```
user@R1>show configuration interface ge-1/0/0
{
   unit 0 {
      family inet {
         filter {
            input RTP-INPUT-FILTER;
         }
         address 10.254.20.1/30;
      }
   }
}
```

show configuration class-of-service interfaces

This command will help you find out what BA classifier, rewrite rules, or scheduler-maps are applied to the interface:

```
user@R1> show configuration class-of-service interfaces
ge-1/0/6 {
   scheduler-map scheduler-map-core;
   unit 0 {
      classifiers {
         dscp DSCP-BA;
         exp EXP-BA;
      }
      rewrite-rules {
         dscp DSCP-BA;
         exp EXP-BA;
      }
   }
}
```

clear interfaces statistics all

This command clears all the interface counters before taking any reading:

```
user@R1> show interfaces ge-2/0/0 extensive
Physical interface: ge-2/0/0, Enabled, Physical link is Up
  Interface index: 134, SNMP ifIndex: 1730, Generation: 137
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, BPDU Error: None, MAC-REWRITE
Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 00:1d:b5:49:b3:fc, Hardware address: 00:1d:b5:49:b3:fc
  Last flapped   : 2010-07-07 07:44:37 PDT (4w6d 11:28 ago)
  Statistics last cleared: 2010-08-10 19:10:48 PDT (00:02:24 ago)
  Traffic statistics:
   Input  bytes  :               112972                 7408 bps
   Output bytes  :               114162                 6848 bps
   Input  packets:                 1610                   13 pps
   Output packets:                 1627                   12 pps
   IPv6 total statistics:
    Input  bytes  :                   0
    Output bytes  :                   0
    Input  packets:                   0
    Output packets:                   0
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :               113198                 7104 bps
   Input  packets:                 1613                   12 pps
   Drop   bytes  :                    0                    0 bps
   Drop   packets:                    0                    0 pps
  Input errors:
   Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Policed discards: 0, L3
incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0, FIFO errors: 0, Resource
errors: 0
  Output errors:
   Carrier transitions: 0, Errors: 0, Drops: 0, Collisions: 0, Aged packets: 0, FIFO
errors: 0, HS link CRC errors: 0, MTU errors: 0, Resource errors: 0
  Ingress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                    0                    0                    0
    1 expedited-fo                   0                    0                    0
    2 assured-forw                   0                    0                    0
    3 network-cont                1613                 1613                    0
  Egress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                    0                    0                    0
    1 expedited-fo                   0                    0                    0
    2 assured-forw                   0                    0                    0
    3 network-cont                1615                 1615                    0
  Queue number:         Mapped forwarding classes
```

```
  0               best-effort
   1                   expedited-forwarding
   2                   assured-forwarding
   3                   network-control
Active alarms  : None
Active defects : None
MAC statistics:                      Receive         Transmit
  Total octets                        112972           114162
  Total packets                         1610             1627
  Unicast packets                       1593             1610
  Broadcast packets                        0                0
  Multicast packets                       17               17
  CRC/Align errors                         0                0
  FIFO errors                              0                0
  MAC control frames                       0                0
  MAC pause frames                         0                0
  Oversized frames                         0
  Jabber frames                            0
  Fragment frames                          0
  VLAN tagged frames                       0
  Code violations                          0
Filter statistics:
  Input packet count                    1610
  Input packet rejects                     0
  Input DA rejects                         0
  Input SA rejects                         0
  Output packet count                                       1627
  Output packet pad count                                      0
  Output packet error count                                    0
  CAM destination filters: 0, CAM source filters: 0
Autonegotiation information:
 Negotiation status: Complete
 Link partner:
    Link mode: Full-duplex, Flow control: Symmetric/Asymmetric, Remote fault: OK
 Local resolution:
    Flow control: Symmetric, Remote fault: Link OK
Packet Forwarding Engine configuration:
 Destination slot: 2
CoS information:
  Direction : Output
  CoS transmit queue                Bandwidth                Buffer Priority   Limit
                              %           bps     %            usec
  0 best-effort              95     950000000    95               0      low    none
  3 network-control           5      50000000     5               0      low    none
  Direction : Input
  CoS transmit queue                Bandwidth                Buffer Priority   Limit
                              %           bps     %            usec
  0 best-effort              95     950000000    95               0      low    none
  3 network-control           5      50000000     5               0      low    none

Logical interface ge-2/0/0.0 (Index 69) (SNMP ifIndex 1733) (Generation 134)
 Flags: SNMP-Traps Encapsulation: ENET2
 Traffic statistics:
  Input  bytes  :               112904
```

```
     Output bytes  :                114024
     Input  packets:                  1610
     Output packets:                  1626
    Local statistics:
     Input  bytes  :                  1156
     Output bytes  :                  1394
     Input  packets:                    17
     Output packets:                    17
    Transit statistics:
     Input  bytes  :                111748                 7408 bps
     Output bytes  :                112630                 6848 bps
     Input  packets:                  1593                   13 pps
     Output packets:                  1609                   12 pps
   Protocol inet, MTU: 1500, Generation: 145, Route table: 0
     Flags: Sendbcast-pkt-to-re
     Addresses, Flags: Is-Preferred Is-Primary
       Destination: 10.254.254.0/30, Local: 10.254.254.2, Broadcast: 10.254.254.3,
Generation: 144
   Protocol multiservice, MTU: Unlimited, Generation: 146, Route table: 0
     Flags: Is-Primary
     Policer: Input: __default_arp_policer__

{master}



user@R1> clear interfaces statistics all

{master}
user@R1> show interfaces ge-2/0/0 extensive
Physical interface: ge-2/0/0, Enabled, Physical link is Up
  Interface index: 134, SNMP ifIndex: 1730, Generation: 137
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, BPDU Error: None, MAC-REWRITE
Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 00:1d:b5:49:b3:fc, Hardware address: 00:1d:b5:49:b3:fc
  Last flapped   : 2010-07-07 07:44:37 PDT (4w6d 11:29 ago)
  Statistics last cleared: 2010-08-10 19:13:54 PDT (00:00:02 ago)
  Traffic statistics:
   Input  bytes  :                 1540                 6160 bps
   Output bytes  :                 1540                 6720 bps
   Input  packets:                   22                   11 pps
   Output packets:                   22                   12 pps
   IPv6 total statistics:
    Input  bytes  :                   0
    Output bytes  :                   0
    Input  packets:                   0
    Output packets:                   0
```

```
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :                 1906                 6064 bps
   Input  packets:                   27                   10 pps
   Drop   bytes  :                    0                    0 bps
   Drop   packets:                    0                    0 pps
  Input errors:
   Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Policed discards: 0, L3
incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0, FIFO errors: 0, Resource
errors: 0
  Output errors:
   Carrier transitions: 0, Errors: 0, Drops: 0, Collisions: 0, Aged packets: 0, FIFO
errors: 0, HS link CRC errors: 0, MTU errors: 0, Resource errors: 0
  Ingress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                    0                    0                    0
    1 expedited-fo                   0                    0                    0
    2 assured-forw                   0                    0                    0
    3 network-cont                  27                   27                    0
  Egress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                    0                    0                    0
    1 expedited-fo                   0                    0                    0
    2 assured-forw                   0                    0                    0
    3 network-cont                  14                   14                    0
  Queue number:       Mapped forwarding classes
    0                 best-effort
    1                 expedited-forwarding
    2                 assured-forwarding
    3                 network-control
  Active alarms  : None
  Active defects : None
  MAC statistics:                       Receive         Transmit
    Total octets                           1540             1540
    Total packets                            22               22
    Unicast packets                          22               22
    Broadcast packets                         0                0
    Multicast packets                         0                0
    CRC/Align errors                          0                0
    FIFO errors                               0                0
    MAC control frames                        0                0
    MAC pause frames                          0                0
    Oversized frames                          0
    Jabber frames                             0
    Fragment frames                           0
    VLAN tagged frames                        0
    Code violations                           0
  Filter statistics:
    Input packet count                       22
    Input packet rejects                      0
    Input DA rejects                          0
    Input SA rejects                          0
    Output packet count                              22
    Output packet pad count                           0
    Output packet error count                         0
```

```
    CAM destination filters: 0, CAM source filters: 0
  Autonegotiation information:
    Negotiation status: Complete
    Link partner:
        Link mode: Full-duplex, Flow control: Symmetric/Asymmetric, Remote fault: OK
    Local resolution:
        Flow control: Symmetric, Remote fault: Link OK
  Packet Forwarding Engine configuration:
    Destination slot: 2
  CoS information:
    Direction : Output
    CoS transmit queue               Bandwidth              Buffer Priority   Limit
                              %            bps     %           usec
    0 best-effort            95      950000000    95              0      low    none
    3 network-control         5       50000000     5              0      low    none
    Direction : Input
    CoS transmit queue               Bandwidth              Buffer Priority   Limit
                              %            bps     %           usec
    0 best-effort            95      950000000    95              0      low    none
    3 network-control         5       50000000     5              0      low    none

  Logical interface ge-2/0/0.0 (Index 69) (SNMP ifIndex 1733) (Generation 134)
    Flags: SNMP-Traps Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                 1540
     Output bytes  :                 1540
     Input  packets:                   22
     Output packets:                   22
    Local statistics:
     Input  bytes  :                    0
     Output bytes  :                    0
     Input  packets:                    0
     Output packets:                    0
    Transit statistics:
     Input  bytes  :                 1540                 6096 bps
     Output bytes  :                 1540                 6648 bps
     Input  packets:                   22                   10 pps
     Output packets:                   22                   11 pps
    Protocol inet, MTU: 1500, Generation: 145, Route table: 0
     Flags: Sendbcast-pkt-to-re
     Addresses, Flags: Is-Preferred Is-Primary
       Destination: 10.254.254.0/30, Local: 10.254.254.2, Broadcast: 10.254.254.3,
Generation: 144
    Protocol multiservice, MTU: Unlimited, Generation: 146, Route table: 0
     Flags: Is-Primary
     Policer: Input: __default_arp_policer__
```

Summary

Careful planning is essential when designing QoS for any network. Once you have gathered all of the requirements then it is only a matter of translating them on the device. The steps illustrated in the previous chapters are symbolic and can be used on any Junos platform including EX, M, MX, and SRX series devices. Also, there are a lot of powerful tools available for troubleshooting, which should aid you in deploying QoS in the network. Finally, Junos CLI is different from the Cisco CLI, but the configuration can be tuned to achieve similar results for QoS.

What to Do Next & Where to Go

http://www.juniper.net/dayone

The *Day One* book series is available for free download in PDF format. Select titles also feature a *Copy and Paste* edition for direct placement of Junos configurations. (The library is available in eBook format for iPads and iPhones from the Apple iBookstore, or download to Kindles, Androids, Blackberrys, Macs and PCs by visiting the Kindle Store. In addition, print copies are available for sale at Amazon or www.vervante.com.)

http://www.juniper.net/books

QoS Enabled Networks: Tools and Foundations, by Peter Lundqvist and Miguel Barreiros. This book, by two experts from Juniper Networks, provides an in-depth treatment of the subject from a more theoretical level all the way through to an understanding of the tools available to influence the behaviors, and finally through to the application of those tools.

http://forums.juniper.net/jnet

The Juniper-sponsored J-Net Communities forum is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions. Register to participate in this free forum.

www.juniper.net/techpubs/

Juniper Networks technical documentation includes everything you need to understand and configure all aspects of Junos, including MPLS. The documentation set is both comprehensive and thoroughly reviewed by Juniper engineering.

www.juniper.net/training/fasttrack

Take courses online, on location, or at one of the partner training centers around the world. The Juniper Network Technical Certification Program (JNTCP) allows you to earn certifications by demonstrating competence in configuration and troubleshooting of Juniper products. If you want the fast track to earning your certifications in enterprise routing, switching, or security use the available online courses, student guides, and lab guides.