

WRED and MDRR on the Cisco 12000 Series Internet Router with a Mix of Unicast, Multicast, and Voice Traffic Configuration Example

Document ID: 22561

Contents

Introduction

Prerequisites

- Requirements
- Components Used
- Conventions

Background Information

Precedence Classes used in the Test

- Coloring of the IP Packets

Expected Results

Network Set-up

3-Port GbE Engine 2 Queuing Implementation

Engine 2 WRED Algorithm in Cisco IOS Software

QoS Configuration

- Rx CoS
- Tx QoS
- Interface Mapping

Starting with No Drops

- Four data streams of 151Mb Each
- Four data streams of 160Mb Each
- Four data streams of 167Mb Each
- Four data streams of 191Mb Each
- Four data streams of 244Mb Each
- All QoS Removed
- Four data streams of 153Mb Each
- Four data streams of 158Mb Each

Adding Ingress Load

- 12-RP#show interfaces g 6/0

Changing the Size of the Streams

Verifying with a 10-Port Gigabit Ethernet Engine 4 Line Card

Related Information

Introduction

This document explains how to configure a Cisco 12000 Series line card for Weighted Random Early Detection (WRED), described in RFC 2309 , in a multi-service environment.

Prerequisites

Requirements

Readers of this document should be knowledgeable of the following:

- Understanding and Configuring MDRR and WRED on the Cisco 12000 Series Internet Router
- How To Read the Output of the show controller frfab | tofab queue Commands on a Cisco 12000 Series Internet Router
- Type of Service in the Internet Protocol Suite, Precedence (RFC–1349)
- Weighted Random Early Detection

Components Used

The information in this document is based on the software and hardware versions below:

- Any Cisco IOS® software release which supports the Cisco 12000 Series Internet Router. Usually these are the 12.0S and 12.0ST releases.
- All the Cisco 12000 platforms are covered by this document. These include the 12008, 12012, 12016, 12404, 12406, 12410, and the 12416.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Background Information

The Cisco 12000 Series is one of the most popular platforms used to build a high bandwidth IP core network. This platform offers the exclusive possibility to configure Quality of Service (QoS).

Since it is more and more common to mix different types of IP traffic (such as Voice over IP – VoIP – and multicast) in the same network, the requirements for prioritization and a controlled dropping behavior become extremely important, and, in many cases, are the difference between success and failure when launching a new service such as VoIP.

The network requirements for different types of IP traffic are beyond the scope of this document. This document focuses on lab tests carried out to find a configuration that is applicable on different line cards, including the Cisco 12000 Series, 3–Port Gigabit Ethernet (3–Port GbE) line card. The results of these tests show that the 3–Port GbE Engine 2 line card is well–suited for a network environment involving a mix of voice, data, and multicast traffic. It also proves that configuring QoS makes a real difference in a congested network.

Precedence Classes used in the Test

The precedence values that are assigned to different classes need to be the same in the entire network. You need to determine a generic policy.

Class	Precedence	Traffic
Evil traffic		All non identified off net traffic (off–net)
On net ---- on net	1	Traffic that stays within the SP network (on–net)
	2	

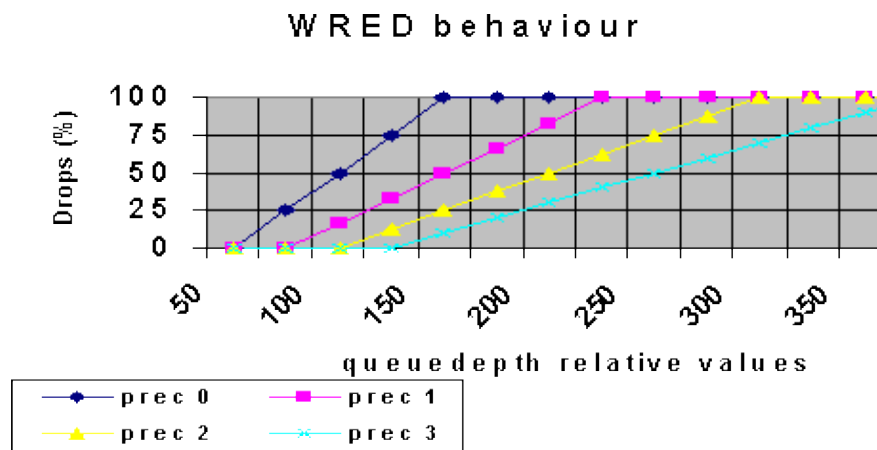
Internet Service Provider (ISP) services		ISP traffic, SMTP, POP, FTP, DNS, Telnet, SSH, www, https
SME (Small and Medium Enterprise)	3	Enterprise customers, a gold service
Real-time, non voice	4	TV, real-time gaming
Voice	5	RTP VOIP traffic
Network control messages	6-7	Border Gateway Protocol (BGP) and other control messages

Coloring of the IP Packets

If QoS is to be implemented in the core of a network, a prerequisite is that the Service Provider be in full control of the precedence value of *all* IP packets transported in the network. The only way to do this is to mark all packets when they enter the network, making no distinction whether they come from the customer/end user side or from the Internet. No marking or coloring should be done in the core.

Expected Results

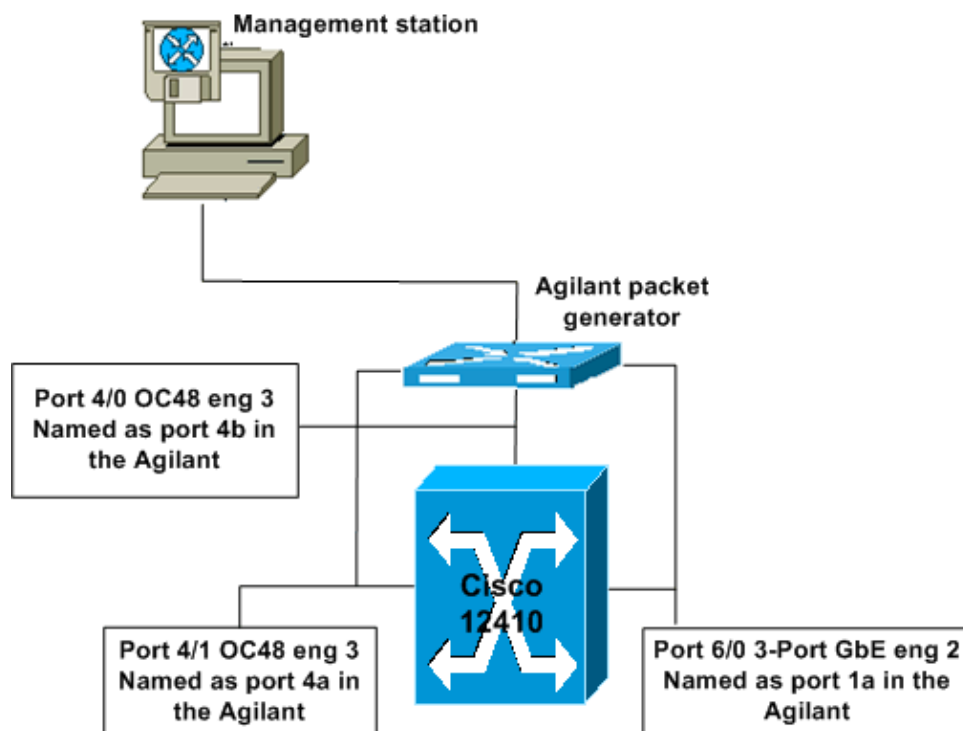
The goal with this design is to have true WRED behavior in classes 0-3. This means that we would like to have a situation where we start dropping precedence 0 packets during congestion. After that, we should also start to drop precedence 1 if the congestion continues, and then also precedence 2 and 3. This is all described in the graph below.



You should have the lowest latency possible for voice packets, and no drops at all for voice and multicast traffic.

Network Set-up

To test and evaluate the configuration, we used a Cisco 12410 together with a packet generator from Agilent. The Cisco 12000 router is running an engineering release based on Cisco IOS Software Release 12.0(21)S1.



3-Port GbE Engine 2 Queuing Implementation

Engine 2 cards normally have eight fromfab queues and one low latency queue, and eight tofab queues per destination slot. There is also a separate tofab multicast queue. On the 3-Port GbE card, there is only one fromfab queue per physical port. In the test, the configuration that was applied specifies more queues. The results show that the 3-Port GbE card accepts this configuration, and the queues are automatically mapped to the queues that are available.

Engine 2 WRED Algorithm in Cisco IOS Software

You must fully understand the algorithm used for WRED in the Engine 2 line card when configuring the minimum and maximum queue-depth values. The code doesn't care about the minimum value configured; instead, it uses its own formula (based on the configured maximum value) to set the minimum value.

Formula:

Minimum value = Maximum value - (the highest power of 2 that doesn't generate a negative result)

The values used in this test resulted in the following minimum values programmed to the Application-Specific Integrated Circuit (ASIC):

Precedence	Configured Minimum	Configured Maximum	Highest Power of 2	Minimum Value in ASIC
0	50	5000	4096	5000 - 4096 = 904
1	60	6000	4096	6000 - 4096 = 1904
2	70	7000	4096	7000 - 4096 = 2904
3	80	8000	4096	8000 - 4096 = 3904

Using this formula to calculate the minimum value means that you can end up with incorrect packet handling behavior if you don't take this into consideration when configuring your WRED parameters. This is shown in the following example:

Precedence	Configured Minimum	Configured Maximum	Highest Power of 2	Minimum Value in ASIC
0	50	150	128	150 - 128 = 22
1	75	225	128	225 - 128 = 97
2	100	300	256	300 - 256 = 44
3	125	375	256	375 - 256 = 119

This means that, even though the values are configured to start dropping according to rule 0 first, then 1, 2 and lastly 3 (above), when the values are written to the ASIC, you actually start dropping precedence 0, then precedence 2, then precedence 1, and lastly precedence 3. There is no way to see which values have been configured in the ASIC on an Engine 2 card. If you apply the configuration on an Engine 3 card, the values showing up in the configuration will be the real values (the recalculated minimum value).

QoS Configuration

For more details about the QoS Configuration, please read Understanding and Configuring MDRR and WRED on the Cisco 12000 Series Internet Router.

Rx CoS

```
rx-cos-slot 2 B2-Table
rx-cos-slot 3 B2-Table
rx-cos-slot 6 B2-Table
```

In most cases, you can use the **rx-cos-slot all** command. In our test case, we had some cards that didn't support tofab queuing, so we couldn't always use the **rx-cos-slot all** command. Instead, we assigned our slot-table to the line cards being used in the test.

```
!
slot-table-cos B2-Table
destination-slot all B2
multicast B2
!--- If you don't fulfill the steps above, you will not be able to
reach the goal of 0 drops for Multicast traffic. With no rx-cos configured,
multicast will be treated in the default queue, meaning it will drop as soon
as there is congestion.
!
```

Tx QoS

Now you may configure your tx-cos. Choose a name for your tx qos, such as "cos-queue-group B2".

Each precedence value that you want to configure a drop behavior for needs to be mapped to a separate random-detect-label.

```
precedence 0 random-detect-label 0
precedence 1 random-detect-label 1
```

```
precedence 2 random-detect-label 2

precedence 3 random-detect-label 3
```

For Modified Deficit Round Robin (MDRR), map each precedence to an MDRR queue. In our example, we mapped precedence 0–3 to the same MDRR queue in order to reserve bandwidth for video (multicast traffic). This mapping provides the requested behavior.

```
precedence 0 queue 0

precedence 1 queue 0

precedence 2 queue 0

precedence 3 queue 0

precedence 4 queue 4
```

Voice is marked with precedence 5, which is why precedence 5 is mapped to the low latency queue.

```
precedence 5 queue low-latency

precedence 6 queue 6

precedence 7 queue 6
```

Now you have to configure the dropping behavior for each of the random-detect-labels. During testing, these numbers were changed until values were found that gave the desired drop pattern. For details, see the Expected Results section. The queue-depth is measured on the physical queue, and not on the MDRR or the RED-Label queue.

```
random-detect-label 0 50 5000 1

random-detect-label 1 60 6000 1

random-detect-label 2 70 7000 1

random-detect-label 3 80 8000 1
```

On the Cisco 12000, it is possible to create Class-Based, Weighted Fair Queuing (CBWFQ) behavior, by giving the different MDRR queue a weight. The default weight is 10 per queue. The number of bytes transmitted every MDRR cycle is a function of the weight value. A value of 1 means 1500 bytes each cycle. A value of 10 means $1500+(9*512)$ bytes per cycle."

```
queue 0 20

queue 4 20

queue 6 20

!
```

Interface Mapping

Each interface needs to be configured for WRED. This is done using the commands:

- **configure terminal**
- **interface gig 6/0**
- **tx-cos B2**

Starting with No Drops

The generated stream uses the following values unless something else is stated:

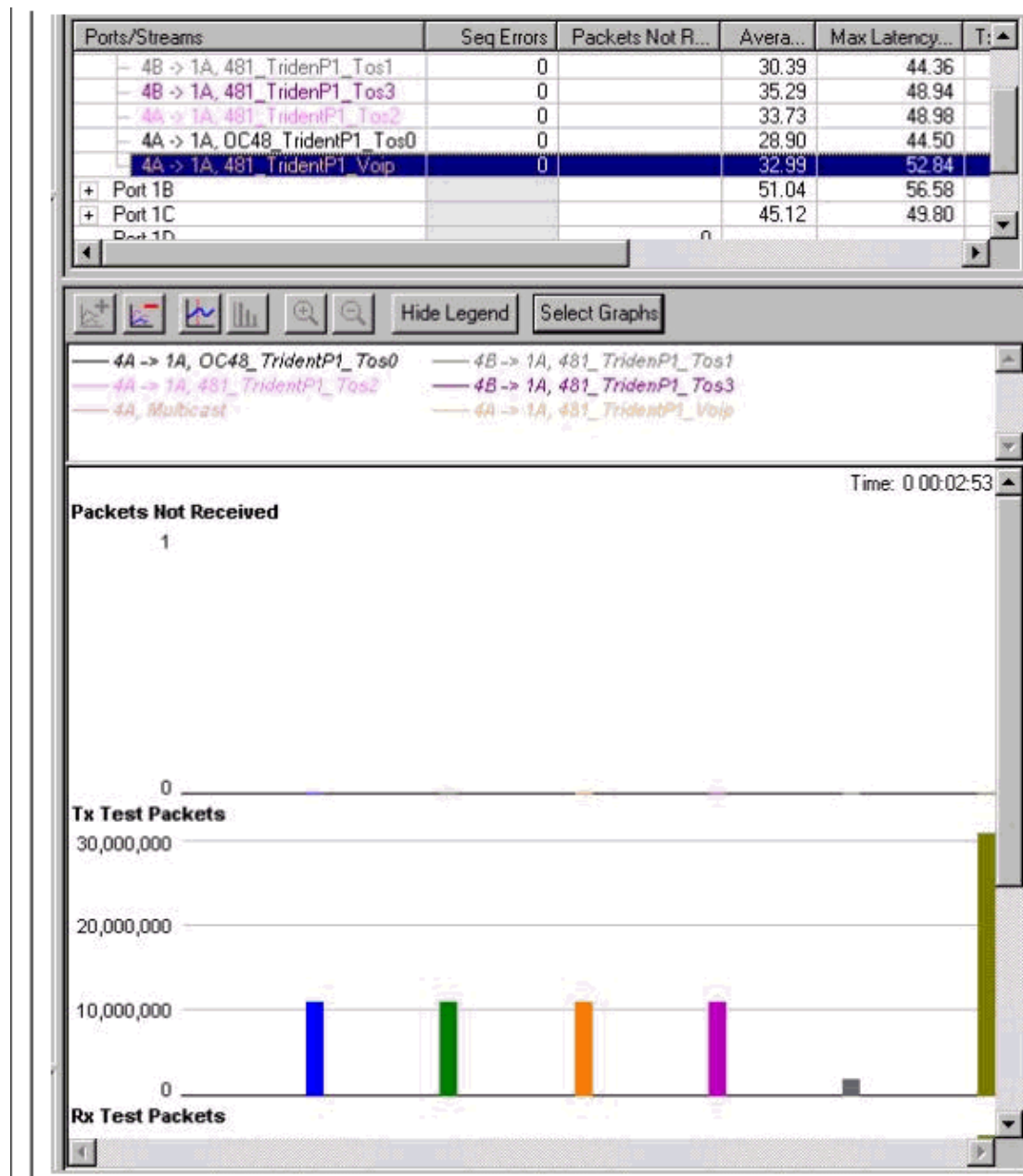
```
MTU all three data streams 300byte, MTU voice 80byte, MTU MC 1500byte  
126Mb MC, 114Mb voip
```

This results in a background stream of 240Mb (VoIP and multicast).

We then add four data-streams of the same size, but with precedence 0-3 (one precedence value per stream).

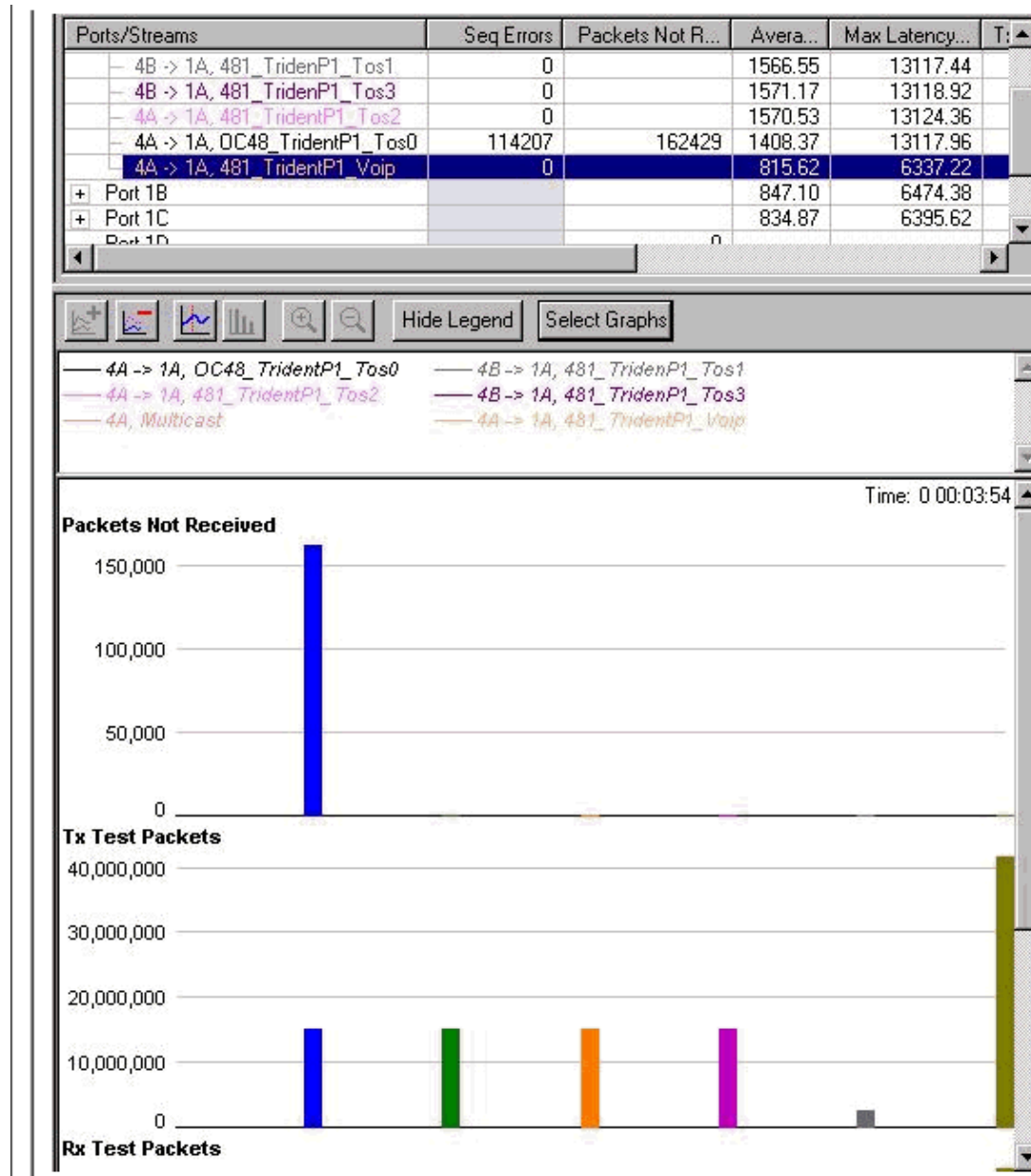
Four data streams of 151Mb Each

This configuration gives a total bandwidth of 844Mb. The graph below shows that there are 0 packet drops, and a very low latency (about 50 us – microseconds – for each stream, including Voice).



Four data streams of 160Mb Each

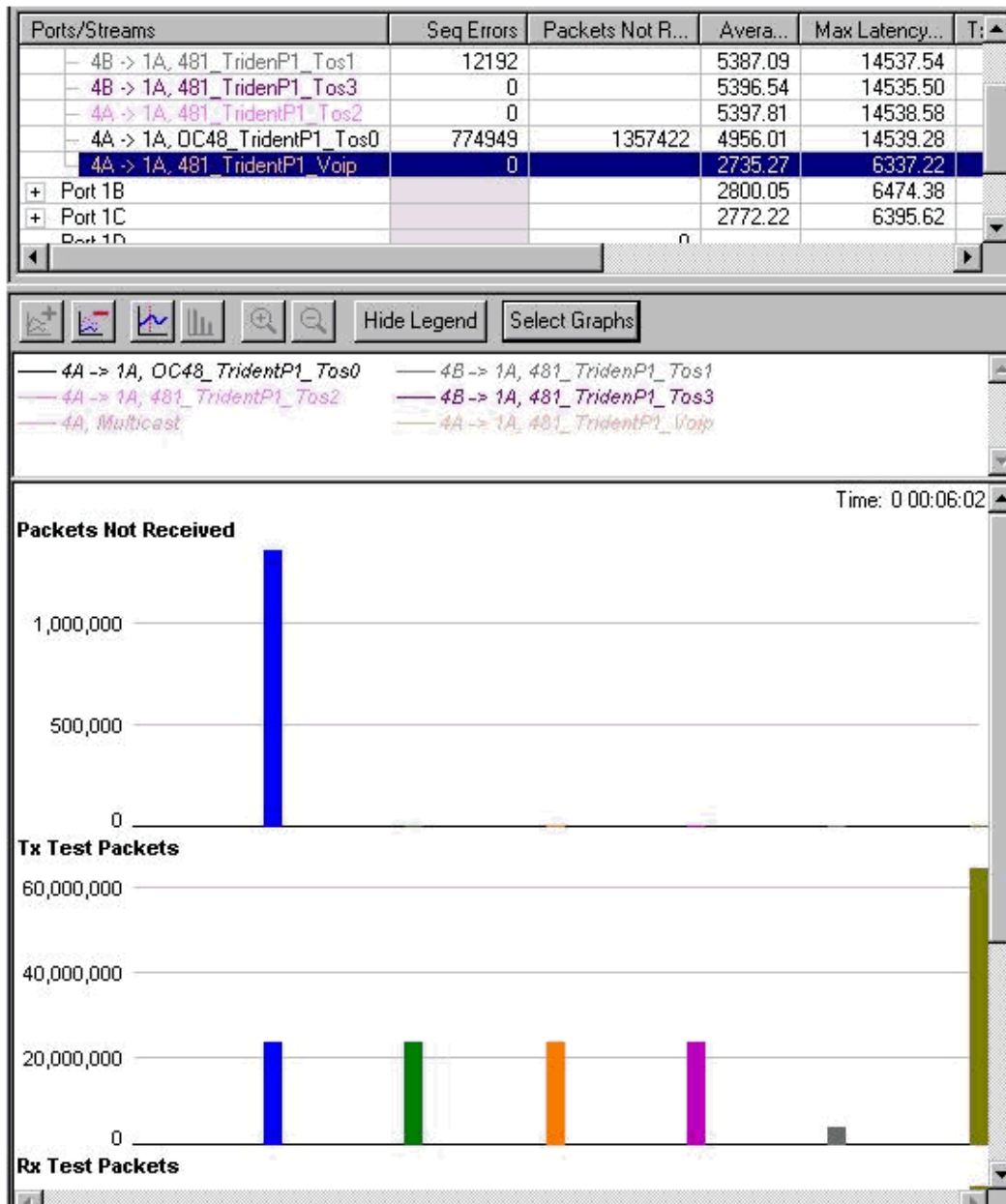
This configuration gives a total bandwidth of 880Mb. The graph below shows that packets start to drop from the precedence 0 class, and the latency for Voice has increased to 6 ms – milliseconds.



Four data streams of 167Mb Each

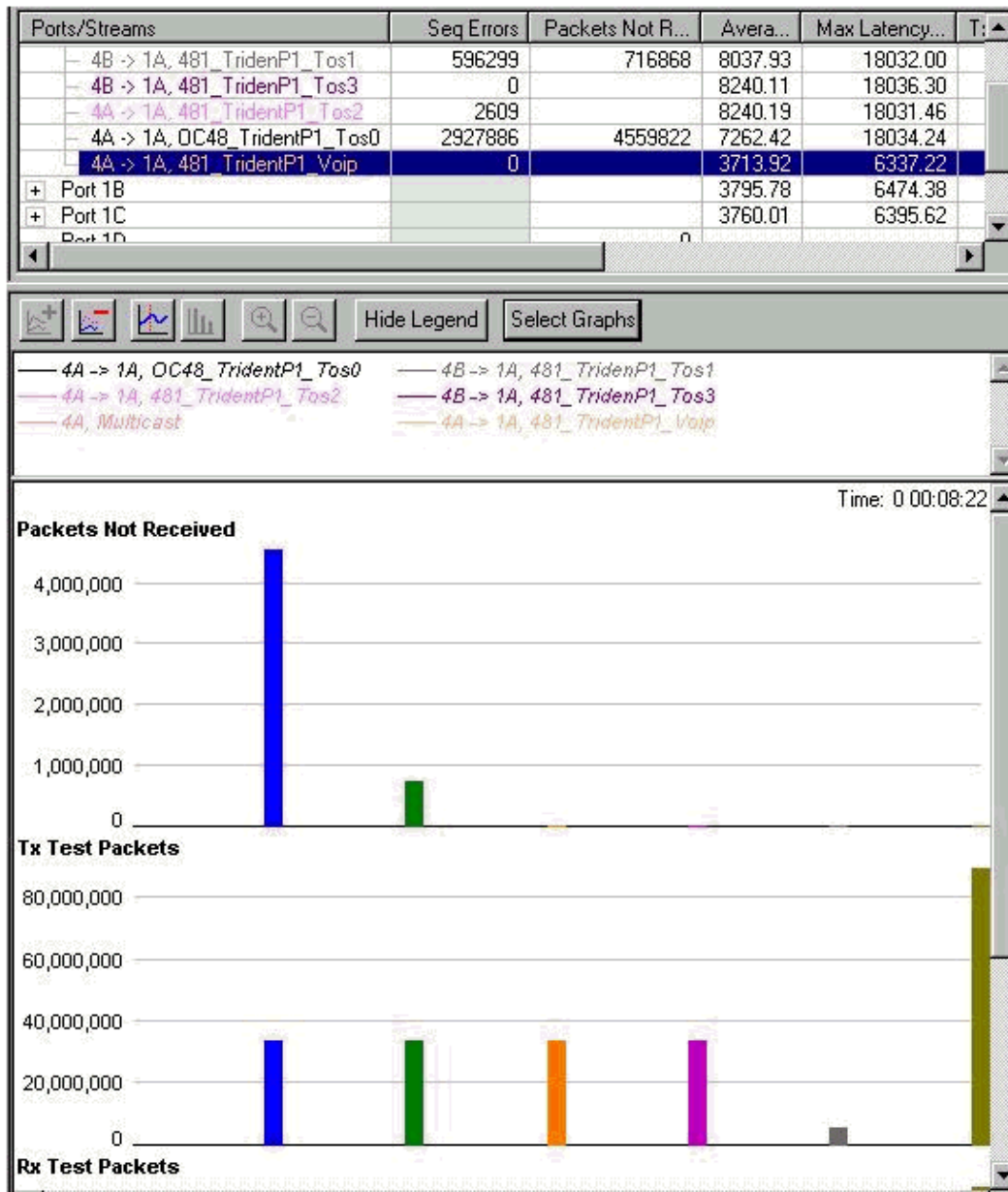
This configuration gives a total bandwidth of 908Mb. Drops are now starting for the precedence 1 class as well. The latency of the Voice traffic is still the same.

Note: The stream was not stopped before being increased, so the difference between the number of drops in stream 0 and 1 is cumulative.



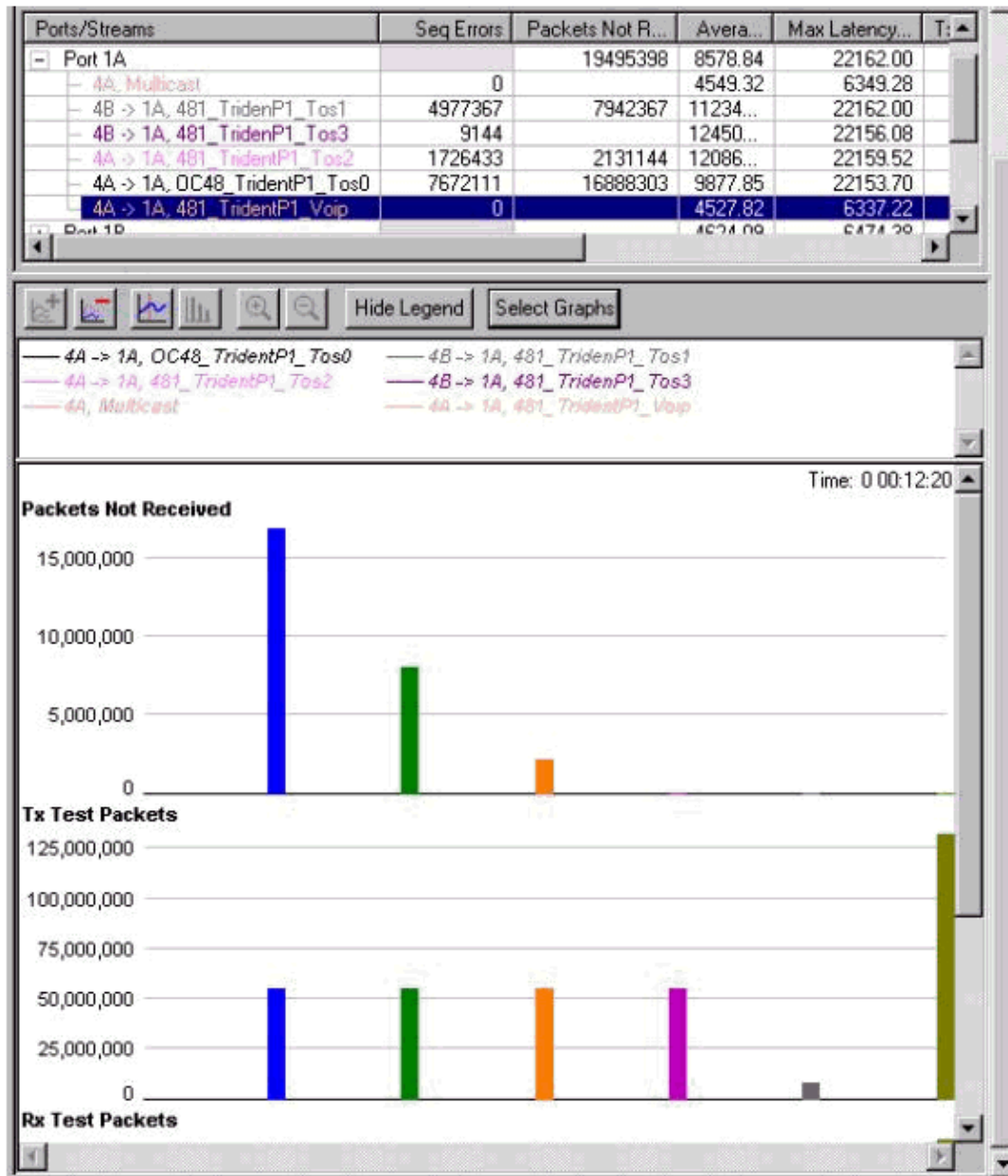
Four data streams of 191Mb Each

When the total bandwidth increases, packets are starting to drop from the precedence 2 queue as well. The total bandwidth we are trying to reach for the Gigabit Ethernet interface is now 1004Mb. This is illustrated in the sequence error counters in the graph below.



Four data streams of 244Mb Each

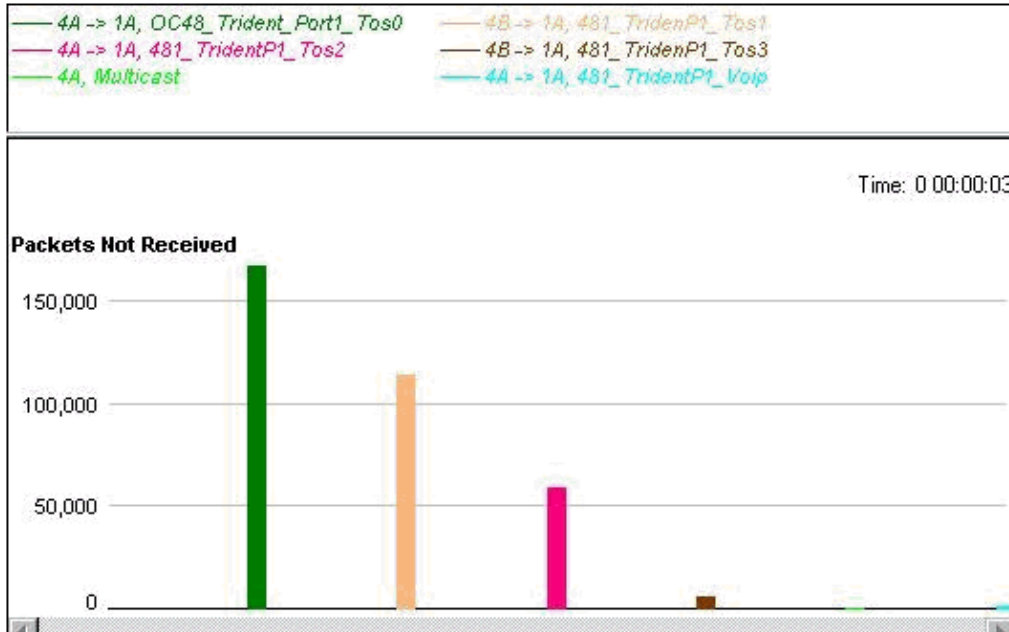
The sequence errors for precedence 3 are starting to increase as well. This is the first sign that drops will start from that queue. The total amount of bandwidth that we are trying to send out the GbE interface is now 1216 Mb. Note that the drops on the multicast (MC) and the Voice queue are still zero, and the latency of the Voice queue has not increased.



Stopping and Starting

All the streams were stopped and started to generate a graph that has cleared counters. This shows how it will look during heavy congestion. As you can see below, the behavior is the desired one.

Ports/Streams	Packets Not Received
All Ports	244885
Port 1A	259941
- 4A -> 1A, OC48_Trident_Port1_Tos0	133621
- 4A -> 1A, 481_TridentP1_Tos2	45024
- 4A -> 1A, 481_TridentP1_Voip	
- 4A, Multicast	
- 4B -> 1A, 481_TridentP1_Tos1	89282
- 4B -> 1A, 481_TridentP1_Tos3	624



All QoS Removed

To prove that QoS really improves performance during congestion, QoS has now been removed and the interface has been congested. The results are below (the generated stream uses the following values unless something else is stated).

MTU all three data streams 300byte, MTU voice 80byte, MTU MC 1500byte

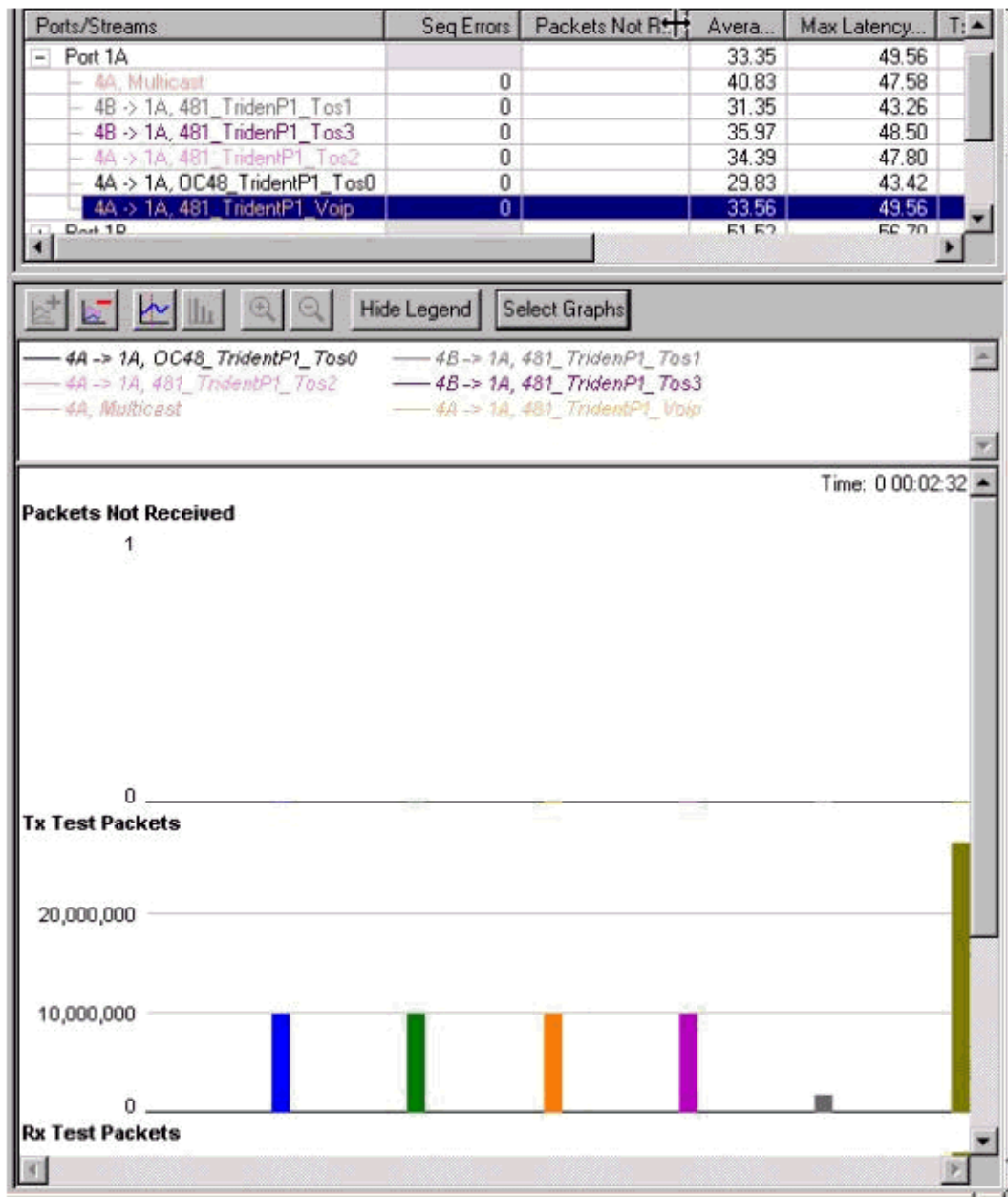
126Mb MC, 114Mb VoIP

This results in a background stream of 240Mb (VoIP and multicast).

We then add four data-streams of the same size, but with precedence 0-3 (one precedence value per stream).

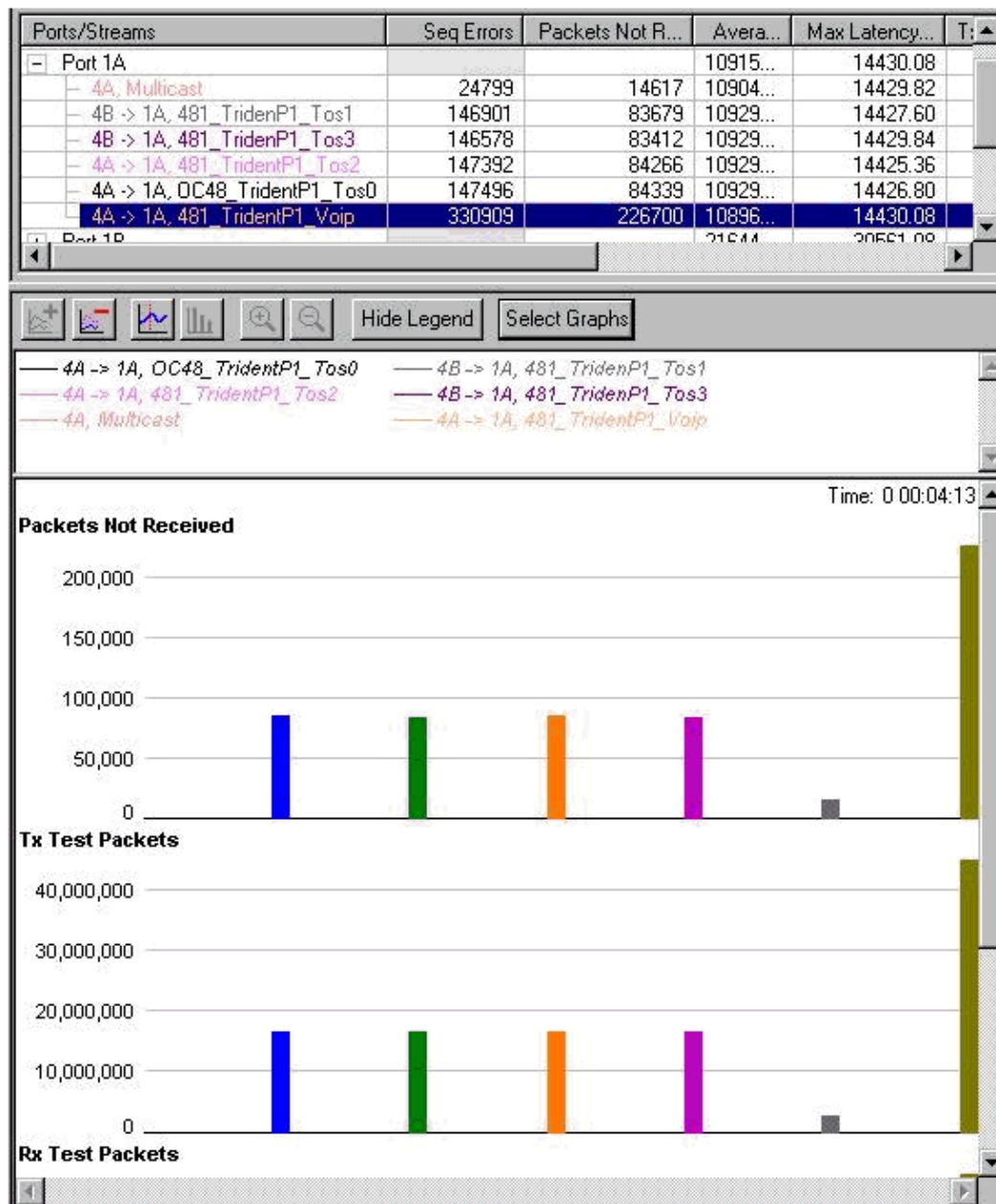
Four data streams of 153Mb Each

This gives a total of 852Mb. There are 0 drops, and a latency of less than 50 us.



Four data streams of 158Mb Each

We start to drop at about the same utilization as when WRED is applied (872Mb). The difference now is that there is a latency of Voice packets of 14 ms (more than twice as much as in the WRED test), and that drops are occurring equally from all classes, including VoIP and multicast.



Adding Ingress Load

So far, all the tests have only included transmitting through the Gigabit Ethernet interfaces. To verify how the interface reacts in a situation where we also congest the interface in the other direction, the following tests were done.

For this test, we loaded the Gigabit Ethernet interface with a total amount of 1056 Mb. This resulted in drops on precedence 0–2, no drops on the precedence 3 traffic. (MC and VOIP remained the same, that is, no drops). We then added traffic in the other direction, as much traffic as the packet generator was able to send out through the Gigabit Ethernet interface. The result is pretty impressive: the receiving congestion doesn't interfere at all with the transmitting queue, and the latency for the receiving traffic is extremely low, less than 13 us for Voice.

- Port 1A	21976403	12050.70	20237.92
- 4A, Multicast		6277.06	6402.88
- 4B -> 1A, 481_TridentP1_Tos1	7729260	16763.90	20232.04
- 4B -> 1A, 481_TridentP1_Tos3		17287.73	20237.92
- 4A -> 1A, 481_TridentP1_Tos2	2090730	17139.98	20233.44
- 4A -> 1A, OC48_TridentP1_Tos0	17210053	16519.08	20236.76
- 4A -> 1A, 481_TridentP1_Voip		6245.21	6376.02
+ Port 1B		6379.18	6512.30
+ Port 1C		6323.52	6432.94
Port 1D	0		
- Port 4A	7	13.09	15.02
- 1A -> 4A, 481_Trident_P3_tos3		13.37	15.02
- 1A -> 4A, 481_TridentP3_Voip		12.67	14.30
- 1A -> 4A, 481_TridentP3_tos2		13.23	14.94
- Port 4B	6	13.11	14.64
- 1A -> 4B, 481_Trident_P3_Tos0		13.11	14.62
- 1A -> 4B, 481_TridentP3_Tos1		13.11	14.64

Ports/Streams	Tx Test Throughput (Mb/s)	Rx Test Throughput (Mb/s)
All Ports		
- Port 1A	747.47	858.75
- 4A, Multicast	126.24	126.24
- 4B -> 1A, 481_TridentP1_Tos1	189.55	142.03
- 4B -> 1A, 481_TridentP1_Tos3	189.54	189.54
- 4A -> 1A, 481_TridentP1_Tos2	189.54	175.70
- 4A -> 1A, OC48_TridentP1_Tos0	189.54	95.24
- 4A -> 1A, 481_TridentP1_Voip	130.00	130.00
+ Port 1B	0.00	126.24
+ Port 1C	0.00	126.24
Port 1D	0.00	0.00
- Port 4A	635.32	424.24
- 1A -> 4A, 481_Trident_P3_tos3	161.62	161.62
- 1A -> 4A, 481_TridentP3_Voip	101.01	101.01
- 1A -> 4A, 481_TridentP3_tos2	161.62	161.62
- Port 4B	379.09	323.23
- 1A -> 4B, 481_Trident_P3_Tos0	161.62	161.62
- 1A -> 4B, 481_TridentP3_Tos1	161.62	161.62

12-RP#show interfaces g 6/0

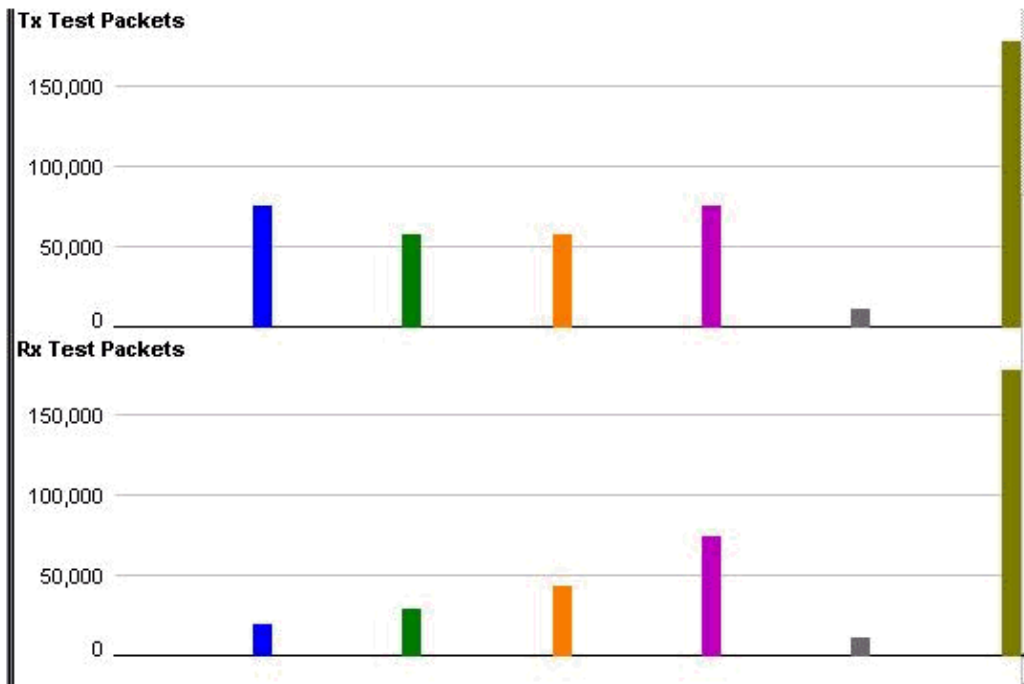
You can monitor the load on the Gigabit link using the **show interfaces** command:

```
Router#show interfaces gig 6/0
GigabitEthernet6/0 is up, line protocol is up
  Hardware is GigMac 3 Port GigabitEthernet, address is 0004.de56.c264
  (bia 0004.de56.c264)
  Internet address is 178.6.0.1/24
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec, rely 255/255, load 232/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex mode, link type is force-up, media type is SX
  output flow-control is unsupported, input flow-control is off
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:05, output 00:00:05, output hang never
  Last clearing of "show interface" counters 08:52:40
  Queueing strategy: random early detection (WRED)
  Output queue 0/40, 2174119522 drops; input queue 0/75, 0 drops
  30 second input rate 838969000 bits/sec, 792079 packets/sec
  30 second output rate 910819000 bits/sec, 464704 packets/sec
    7584351146 packets input, 1003461987270 bytes, 0 no buffer
      Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
        0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
        0 watchdog, 514 multicast, 0 pause input
      11167110605 packets output, 2241229569668 bytes, 0 underruns
        0 output errors, 0 collisions, 0 interface resets
```

```
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier, 0 pause output
0 output buffer failures, 0 output buffers swapped out
```

Changing the Size of the Streams

To verify that the test results are not due to the bandwidth being the same for all the streams, we changed the streams so that they were transmitting different amounts of data. We also tried to change the maximum transmission unit (MTU) so it was different for each stream. With the configured queue values, the result was still the same, dropping precedence 0 first, then 1, then 2, and lastly precedence 3.



Verifying with a 10-Port Gigabit Ethernet Engine 4 Line Card

Since the latency of the VoIP queue (the Low latency queue) in the test was fairly high, we performed the same test with the 10-Port Gigabit Ethernet Engine 4 line card. As expected, the result with this line card was much better regarding latency in the low latency queue (LLQ). The results were the same with regards to how the dropping occurred. The latency for the LLQ was around 10us, which is 1:1000 of the latency in the 3-Port Gigabit Ethernet Engine 2 line card.

Related Information

- [Understanding and Configuring MDRR and WRED on the Cisco 12000 Series Internet Router](#)
- [How To Read the Output of the show controller frfab | tofab queue Commands on a Cisco 12000 Series Internet Router](#)
- [Type of Service in the Internet Protocol Suite, Precedence \(RFC-1349\)](#)
- [Weighted Random Early Detection](#)
- [Technical Support & Documentation – Cisco Systems](#)

