Southern Polytechnic State University

Spring Semester 2009

ECET 4840 Laboratory Exercises 9: Router Queuing Configuration and Testing

Objective:

Students will investigate the effect of varying link capacity and queuing mechanisms on a Cisco 261 XM router's ability to handle relatively large volumes of data traffic.

Background:

Routers perform queuing when the arrival rate exceeds the departure rate of data traffic. In other words when a router switches data to an output interface faster than data arrives no queuing occurs even if the interface has been configured for queuing. Such queuing is configured on an interface by using a service policy. Although others exist we will discuss four queuing schemes: FIFO (First In First Out), FQ (Fair Queuing), WFQ (Weighted Fair Queuing), and CBWFQ (Class-Based WFQ); FQ is not available for Cisco routers.

In FIFO the first bit of a packet determines its order in a queue. Therefore, if a large packet arrives ahead of a smaller packet, then the smaller packet must wait for the larger packet to be fully queued and transmitted prior to selection by the interface. In high-speed interfaces (> 2Mbps) problems associated with FIFO queues are minimized since the packet transmission rate is relatively high. However, in lower speed queues problems such as packet jitter and delay may become more pronounced.

In FQ smaller packets, which are queued up faster that larger ones, are allowed to be transmitted while the larger packets are still arriving. This type of queuing address the inherent delay associated with FIFO queues, but it does not have provisions for traffic prioritization for varying types of data flows (referred to as conversations), such as a VoIP flow versus the downloading of a webpage.

WFQ sorts arriving packets according to the flow type with which they are associated into individual FIFO queues. Queued packets are "weighted" based on QoS (Quality of Service) bits found in the packets' header. In other words packets possessing a higher QoS receive a higher transmission priority than packets that do not. WFQ is the default queuing scheme used on low-speed interfaces on Cisco routers.

CBWFQ is a superset of WFQ providing additional support for user-defined classes of traffic essentially enabling the router administrator to define the conversations. Packets are “weighted” according to class per the amount of bandwidth assigned to that class. CBWFQ is supported in higher-speed interfaces, such as DS3, in Cisco routers.

Overview:

You will work in small teams of 2 or 3 people. Each team will conduct traffic throughput experiments between Fast-Ethernet ports on a specific router (**lab 8**) and via the serial link between two routers (**lab 9: next week’s lab**) (see Figure 1).

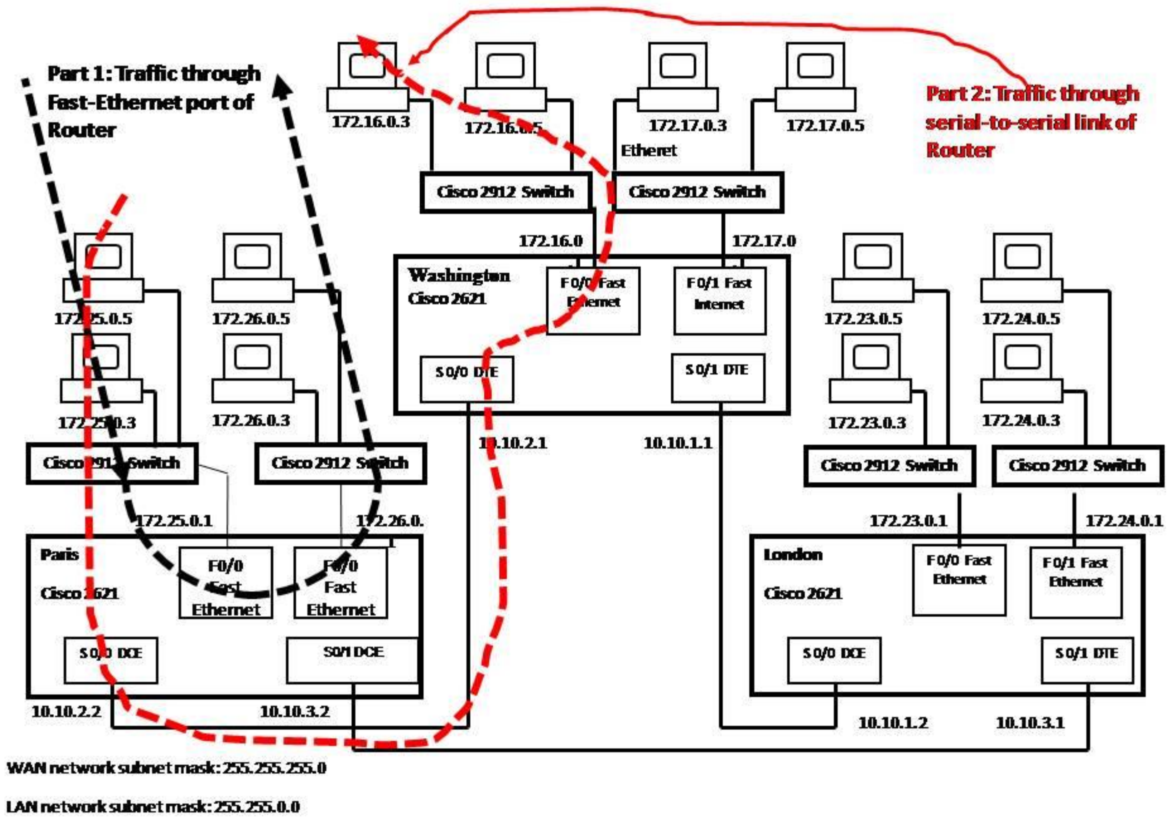


Fig. 1. Cisco 2621 Router Intranet Configuration

You will perform the following tasks:

1. Configure the network depicted in the Figure. Note the use of separate switches to connect the workstations to the router. If needed, a switch can be replaced by a hub. **Note 2: You will have to coordinate with other teams at various times during the lab in order to properly collect your required data.**
2. Use Iperf and wireshark to generate and analyze data traffic transmitted through a Fast_Ethernet-Fast_Ethernet-based path with default queuing and capacity settings (lab 8).
3. Use Iperf and wireshark to generate and analyze data traffic transmitted through a Fast_Ethernet-serial-link Fast_Ethernet-based path with varying queuing and capacity settings (lab 9: next week's lab).
4. Generate appropriate plots and tables relevant to the following procedures.

Procedure:

Disconnect the Lab from the Campus Network

1. Ask your instructor to disconnect the lab from the campus network.
2. Log into to your team's router by using the following command at a Windows command prompt on the designated client workstation: ***telnet avocent_ip_address router_port_#***

   Note: *avocent_ip_address* corresponds to the actual IP address of the Avocent device and *router_port_#* corresponds to the router's port number listed in the Avocent sheet in the lab (e.g. 3005).
3. Use IPerf on the designated client and server workstations:

   In the following steps you will work with a TCP connection. Note: your traffic must flow through the router Fast-Ethernet ports. Verify connectivity between the chosen client and server workstations using ping and/or by viewing the router's routing table using: ***sh ip route***

   a) Open a command prompt on each machine and move to the **c:\program files\iperf** folder.

   b) On the server computer type **iperf –s** .

      Important: when you need to exit the server process, hit Ctrl-C (both keys at the same time).

   c) On the client computer type **iperf –c *ipaddress*** . Where ***ipaddress*** is the address of the server computer. If all is well, the utility will run successfully and calculate the payload data rate in bits per second.

d) Run Wireshark and set up a capture with a filter to extract only traffic to/from your machines and adjust the buffer to 10 Mbytes.

e) Determine how to change the length of time that the traffic is generated, change it to 20 seconds, and repeat your test. The iperf documentation is in the **index.html** file in the iperf applications **doc** subfolder.

f) Examine the packet payloads.

   i) What is the size of the payload?

   ii) Are there any distinctive features of the payload?

   iii) Examine the **IO Graph** in the **Statistics** menu and compare the traffic levels in bps from the iperf display and that of the graph. **Make note of the throughput.**

      Throughput: ____

4. Now you will send constant-bit-rate UDP streams between the two computers and examine the throughput and jitter. The server and client commands and listed below (for a tutorial see: http://www.openmaniak.com/iperf.php). We will be using a packet size of 400 bytes.

iperf –s –u –w 128k –l 400 –i 1	iperf –c *serveraddr* –u –w 128k –l 400–b 80m

Important: Make sure that the buffer size (-w) and datagram length (-l) match in the client and server commands.

a) Open a command prompt window on the server and client computers and migrate to the **c:\program files\iperf\** directory on each.

b) Open Wireshark (on either the client or server or both) and set up a capture with a filter to extract only traffic to/from your machines and adjust the buffer to 10 Mbytes. Before starting your capture, you need to disable the **Redback** dissector. This dissector will generate many confusing error messages in the **Packet List Pane**. To disable, click **Analysis→Enabled Protocols.** In the **Enabled Protocols** box, scroll down until you see **Redback** and **RedbackLI** and <u>uncheck</u> them. Click **OK.**

   Do not start your capture yet.

c) Enter the first server command on the server and press **Enter**. The server process will start and the cursor will vanish.

d) Enter the first client command on the client but do not press **Enter**. Start your Wireshark capture and then press **Enter** to start the client process. The results will be displayed on the server computer. Once the server displays its final output, stop the capture.

e) Save or capture the information displayed on the server. Make note of the measured bandwidth and the jitter.

g) Display the following three Wireshark IO Graph of the captured packet buffer. **Save a copy of each for your report. There are some questions in the questions section that apply to the graphs. You should check the questions before completing this section.**

   i) Set the vertical units to **packets/tick**, **Auto** and the horizontal **tick interval** to 0.001 seconds. Save the plot.

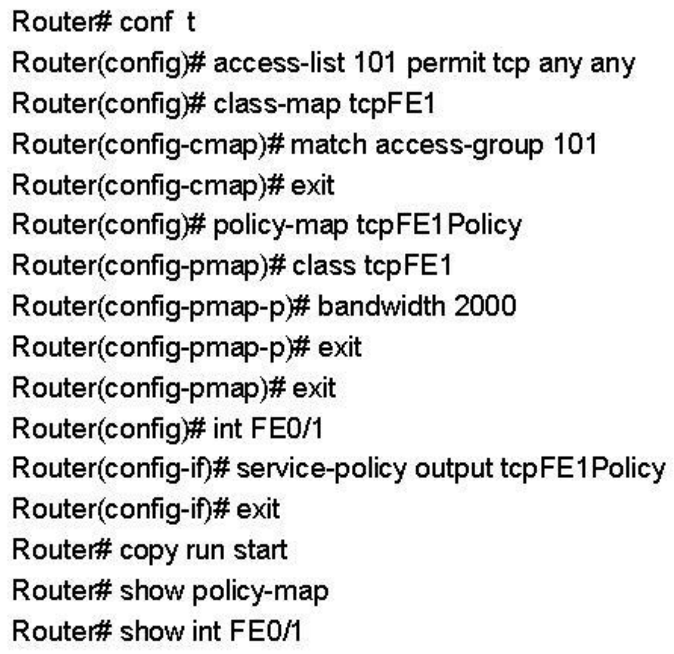
   ii) Change to **packets/tick** and a 1 second **tick interval**. Save the plot.

   iii) Change to **bits/tick** and a 1 second **tick interval.** Save the plot.

5. We will not modify our Fast_Ethernet queuing configuration; the default is FIFO queuing. However, when a service policy is applied the queuing changes to CBWFQ.

   Important: CBWFQ requires three configuration components: i) the class-map which groups similar conversations (flows) into classes, ii) the policy-map specifies how each conversation is to be handled (e.g. bandwidth allocation), and iii) the service-policy which associates a policy-map with an interface.

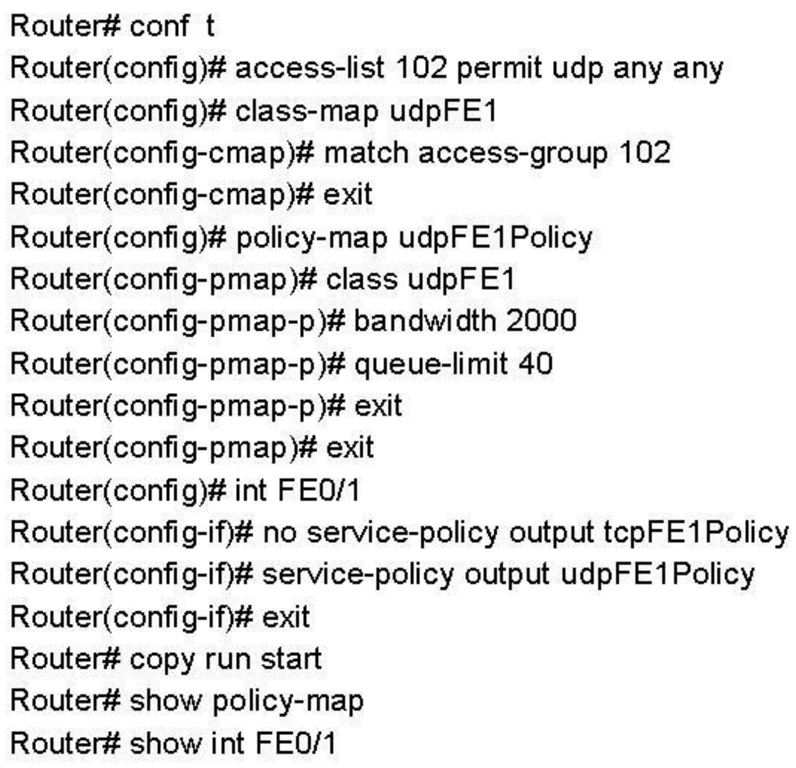
6. Using your telnet window configure your router to use CBWFQ by entering (and reviewing) the following configuration additions:



```
Router# conf  t
Router(config)# access-list 101 permit tcp any any
Router(config)# class-map tcpFE1
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit
Router(config)# policy-map tcpFE1Policy
Router(config-pmap)# class tcpFE1
Router(config-pmap-p)# bandwidth 2000
Router(config-pmap-p)# exit
Router(config-pmap)# exit
Router(config)# int FE0/1
Router(config-if)# service-policy output tcpFE1Policy
Router(config-if)# exit
Router# copy run start
Router# show policy-map
Router# show int FE0/1
```

7. Repeat steps 1 through 3. Recall that procedures 1 through 3 used tcp to perform data throughput calculations. Access-list 101 sets up a means by which we can classify tcp traffic to be handled via our queuing policy.

8. Repeat steps 1 through 3 after changing the bandwidth parameter to 1000.
9. Using your telnet window change your router's CBWFQ configuration to:



```
Router# conf  t
Router(config)# access-list 102 permit udp any any
Router(config)# class-map udpFE1
Router(config-cmap)# match access-group 102
Router(config-cmap)# exit
Router(config)# policy-map udpFE1Policy
Router(config-pmap)# class udpFE1
Router(config-pmap-p)# bandwidth 2000
Router(config-pmap-p)# queue-limit 40
Router(config-pmap-p)# exit
Router(config-pmap)# exit
Router(config)# int FE0/1
Router(config-if)# no service-policy output tcpFE1Policy
Router(config-if)# service-policy output udpFE1Policy
Router(config-if)# exit
Router# copy run start
Router# show policy-map
Router# show int FE0/1
```

10. Repeat step 4. Note: since our queuing policy uses a 2000 Kbps restriction be sure to change the iperf client bandwidth parameter from “-b 80m” to “-b 2m”.
11. Repeat step 4 after changing the queue-limit parameter to 15. Note: this parameter specifies the number of packets allowed in the queue.
12. Repeat step 4 after changing the queue-limit parameter to 4.

Questions:

1. Compare the three UDP iperf IO Graphs and discuss how the various queuing modifications affected the results.
2. Compare the throughput data from the TCP and three UDP tests and how they were affected by the queuing modifications.
5. Compare the jitter levels in the UDP tests and explain how the queuing modifications affected them.
6. Explain how queue-limit affects UDP throughput and jitter.

Conclusions:

Bring together the concepts illustrated in the lab and make conclusions as to what they mean, and what you learned.

Turn in a report containing the following: a cover sheet, introduction, screenshots, answers to all questions, and a conclusion. Staple your report in the upper left hand corner.

References:

http://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/cbwfq.html

http://www.cisco.com/en/US/docs/ios/12_4t/12_4t4/ht_fpm.html#wp1083370

http://www.informit.com/articles/article.aspx?p=102233&seqNum=4

http://wiki.nil.com/Queuing_Principles_in_Cisco_IOS/Fair