Custom Queuing

Custom queuing is a platform-independent congestion management mechanism very similar to priority queuing. Custom queuing provides fairness in servicing the queues, unlike priority queuing, which can potentially starve non–high priority queues. Going back to the grocery store line example, let's say the clerk can now service only up to a certain number of customers from the club card line before returning to the customers in the non–club card line, and vice-versa. The clerk now alternates between the two lines, checking out the number of people allowed in one line before switching to the other line.

Custom queuing allows 16 configurable queues, each with a count specifying how many bytes of data should be delivered from the queue before IOS moves to the next one. In other words, the 16 queues are serviced in a round-robin fashion, and a configured number of bytes are serviced from each queue before moving to the next queue.

NOTE

There are actually 17 queues in IOS custom queuing but the user cannot classify packets into queue 0. Queue 0 is a special queue reserved for control traffic, such as interface keepalives and routing updates.

Let's use the configuration in Example 8-4 and work through a short example.

Example 8-4. Custom Queuing Example

access-list 110 permit ip host 10.1.1.1 any access-list 120 permit ip host 172.16.28.1 any ! queue-list 1 protocol ip 1 list 110 queue-list 1 protocol ip 2 list 120 queue-list 1 default 3 ! queue-list 1 queue 1 byte-count 1000 queue-list 1 queue 2 byte-count 1000 queue-list 1 queue 3 byte-count 2000 ! interface Serial 0 custom-queue-list 1

Assume IOS receives the following packets for switching:

Packet 1—

Sourced from 172.18.40.8, length 100 bytes

Packet 2—

Sourced from 10.1.1.1, length 500 bytes

Packet 3—

Sourced from 172.18.40.8, length 100 bytes

Packet 4—

Sourced from 10.1.1.1, length 1000 bytes

Packet 5—

Sourced from 172.16.28.1, length 200 bytes

Packet 6—

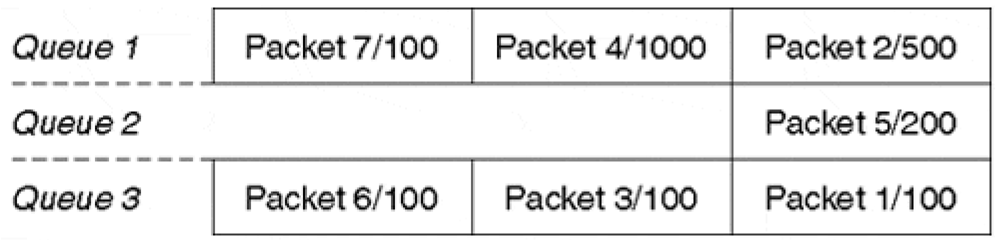
Sourced from 172.18.40.8, length 100 bytes

Packet 7—

Sourced from 10.1.1.1, length 100 bytes

After IOS receives and classifies these seven packets, the queues look like Figure 8-3.

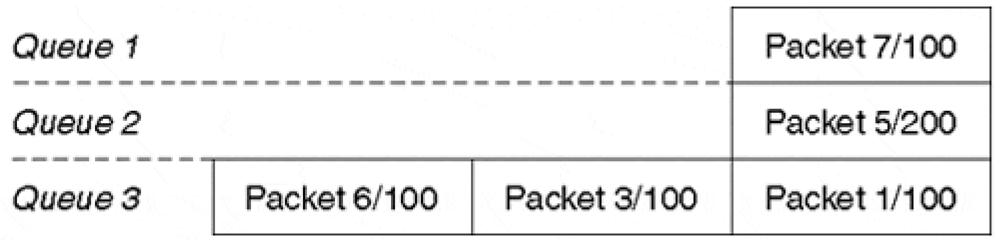
Figure 8-3. Initial Packet Distribution for Custom Queuing Example



IOS begins with queue 1; the first packet on the queue is removed and transmitted. This would be packet 2, which is 500 bytes long. The size of packet 2 is subtracted from the allowed byte count for this queue, 1000, leaving a remainder of 500. Because the remainder is greater than 0, the next packet, packet 4, is removed from queue 1 and is transmitted.

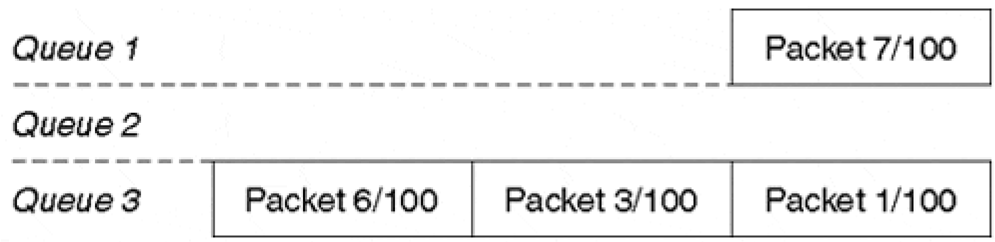
After transmitting packet 4, queue 1 has less than 0 bytes in its allowed byte count. So, IOS moves on to service queue 2. Figure 8-4 shows the queues after this first stage of processing.

Figure 8-4. Packet Distribution after Queue 1 Is Serviced the First Time



Next, packet 5 is transmitted from queue 2 and its length subtracted from the total bytes allowed for this queue, 1000, leaving 800 bytes. Another packet could be transmitted from this queue, but no packets remain to transmit. Figure 8-5 illustrates the queues after this second stage of processing.

Figure 8-5. Packet Distribution after Queue 2 Is Serviced



Finally, we come to the last queue, queue 3. Packet 1 is transmitted and its length, 100 bytes, is subtracted from the total byte count of 2000 allowed for this queue. The remaining two packets, 3 and 6, also are transmitted. Finally, the router returns to queue 1 and transmits the final packet.

As you can see, custom queuing doesn't allow any queue to starve; it always services every queue at some point. If a queue fills up faster than it can be serviced, packets tail drop when the number of packets in the queue exceeds the maximum depth of the queue. Thus, custom queuing in effect provides a minimum bandwidth allocation for each configured type of traffic during periods of congestion, but doesn't guarantee delivery above that minimum bandwidth. Packets are delivered above that minimum bandwidth if the other queues are not congested. In this example, Queue 1 is guaranteed to be able to transmit at least 1000 bytes out of every cycle of about 4000 bytes. The end result of this is that Queue 1 is guaranteed to be able to utilize at least 25 percent of the interface bandwidth during periods of congestion.

NOTE

A queue can completely transmit its last packet of that cycle if there is at least 1 byte left in its byte count.

The **show queueing custom** command provides information on which queues are configured and how they are configured, as demonstrated in Example 8-5.

Example 8-5. *show queueing custom* Displays Information about Which Queues Are Configured and How They Are Configured

router#**show queueing custom** Current custom queue configuration: List Queue Args 1 3 default 1 1 protocol ip list 110 1 2 protocol ip list 120 1 1 byte-count 1000 1 2 byte-count 1000 1 3 byte-count 2000

Again, this output doesn't provide any information on how many packets are waiting in each queue or how many packets were dropped from any queue. For this information, you need to look at **show interface** as shown in Example 8-6.

Example 8-6. *show interface* Displays Packet Traffic through a Queue

router#**show interface serial 0** Serial0 is up, line protocol is up …. Queueing strategy: custom-list 1 Output queues: (queue #: size/max/drops) 0: 0/20/0 1: 0/20/0 2: 0/20/0 3: 0/20/0 4: 0/20/0 5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0 10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0 15: 0/20/0 16: 0/20/0 ….

From the output in Example 8-6, you can see the type of queuing listed (under **Queueing strategy**), the current number of packets in each queue, the maximum number of packets allowed in each queue, and the number of packets dropped out of each queue.

Last updated on 12/5/2001

< BACK   Make Note | Bookmark   CONTINUE >

Index terms contained in this section