CLSC Book

Switching Paths

The basic cisco switching paths are Process Switching, Fast Switching, Optimum Switching, Distributed Switching, and NetFlow Switching. Through the switching process, the router determines the next hop toward the destination address. Switching moves traffic from an input interface to one or more output interfaces. Switching is optimized and has lower latency than routing because it can move packets, frames, or cells from buffer to buffer with simpler determination of the source and destination of the traffic. It saves resources because it does not involve extra lookups.

Process Switching

In process switching the first packet is copied to the system buffer. The router look up the Layer 3 network address in the routing table and initializes the fast-switch cache. The frame is rewritten with the destination address and sent to the exit interface that services that destination. Subsequent packets for that destination are sent by the same switching path. The route processor computes the cyclical redundancy check (CRC).

Fast Switching

Fast switching is enabled by default on all interfaces that support fast switching. If you have a situation where you need to disable fast switching and fall back to the process-switching path, understanding how various processes affect the router and where they occur will help you determine your alternatives. This is especially true when you are troubleshooting traffic problems or need to process packets that require special handling. Some diagnostic or control resources are not compatible with fast switching or come at the expense of processing and switching efficiency. Understanding the effects of those resources can help you minimize their effect on network performance.

When packets are fast switched, the first packet is copied to packet memory and the destination network or host is found in the fast-switching cache. The frame is rewritten and sent to the exit interface that services the destination. Subsequent packets for the same destination use the same switching path. The interface processor computes the CRC.

Optimum Switching

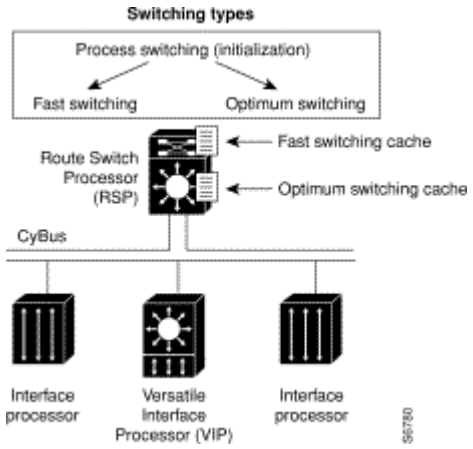
Optimum switching is similar to fast switching, but is faster. The first packet is copied to packet memory and the destination network or host is found in the optimum-switching cache. The frame is rewritten and sent to the exit interface that services the destination. Subsequent packets for the same destination use the same switching path. The interface processor computes the CRC.

Optimum switching is enabled by default on Cisco 7500 series routers; it must be disabled for debugging.

Distributed Switching

Switching becomes more efficient the closer to the interface the function occurs. In distributed switching, the switching process occurs on VIP and other interface cards that support switching. For model numbers and hardware compatibility information, refer to the Cisco Product Catalog. Figure 6 illustrates the distributed switching process on the Cisco 7500 series.

Distributed Switching on Cisco 7500 Series Routers



The VIP card installed in this router maintains a copy of the routing cache information needed to forward packets. Because the VIP card has the routing information it needs, it performs the switching locally, making the packet forwarding much faster. Router throughput is increased linearly based on the number of VIP cards installed in the router.

NetFlow Switching

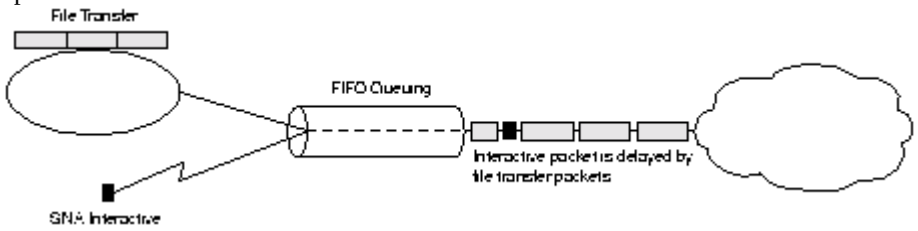
NetFlow switching enables you to collect the data required for flexible and detailed accounting, billing, and chargeback for network and application resource utilization. Accounting data can be collected for both dedicated line and dial-access accounting. NetFlow switching over a foundation of VLAN technologies provides the benefits of switching and routing on the same platforms. NetFlow switching is supported over switched LAN or ATM backbones, allowing scalable inter-VLAN forwarding. NetFlow switching can be deployed at any location in the network as an extension to existing routing infrastructures. NetFlow switching is described in "Configuring NetFlow Switching" later in this publication.

Queuing Algorithms

The Cisco IOS software implements four different output queuing algorithms: First in, first out queuing (FIFO), Priority queuing, Custom queuing, and Weighted fair queuing Each queuing method has advantages and

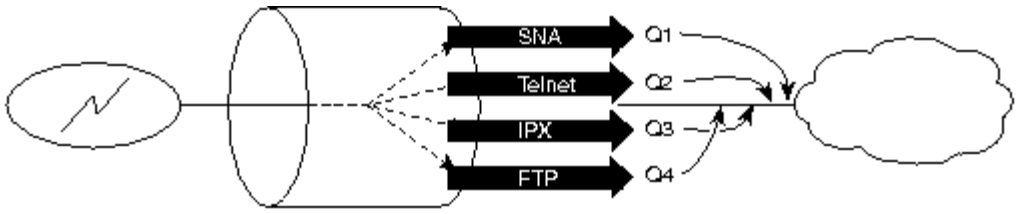
First In, First Out Queuing
This is the simplest and most common interface queuing technique and works well if links are not congested. The first packet to be placed on the output interface queue is the first packet to leave the interface. The problem with first in, first out queuing is that when a station starts a file transfer, it can consume all the bandwidth of a link to the detriment of interactive sessions. The phenomenon is referred to as a packet train because one source sends a "train" of packets to its destination and packets from other stations get caught behind the train. First in, first out queuing is effective for large links that have little delay and minimal congestion.

Potential Impact of a File Transfer on Interactive Traffic



Priority Queuing

Priority queuing allows network managers to define how they wish traffic to be prioritized in the network. By defining a series of filters based on packet characteristics, traffic is placed into a number of queues; the queue with the highest priority is serviced first, then the lower queues are serviced in sequence. If the highest priority queue is always full, then this queue will continually be serviced and packets from the other queues will queue up and be dropped. In this queuing algorithm one particular kind of network traffic can dominate all others. Priority queuing assigns traffic to one of four queues: high, medium, normal, and low.




```
Interface Serial1
ip address 20.0.0.1 255.0.0.0
priority-group 1
!
priority-list 1 protocol ip high tcp 2065
priority-list 1 protocol ip medium tcp 23
priority-list 1 protocol ipx normal
priority-list 1 protocol ip low tcp 21
```

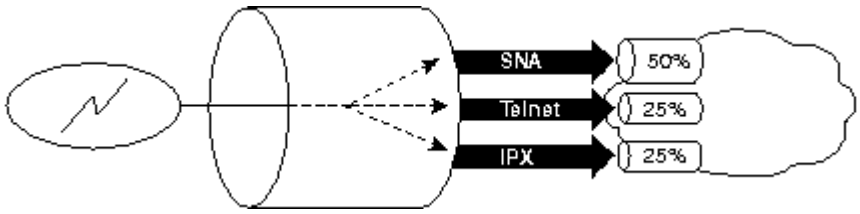
Priority Queuing Services Traffic on the Highest Priority Queue First

In picture above the priority-group command assigns priority list 1 to Serial1. The priority-list command defines the queuing algorithm to be used by queue list 1 and maps the traffic into various queues. Priority queuing is useful when you want to guarantee that the DLSw+ traffic will get through even if it delays other types of traffic. It works best if the DLSw+ traffic is low volume (for example, a small branch with a transaction rate of five to ten transactions per minute), and the number of queues is kept to a minimum (two or three). In this configuration, DLSw+ is in the highest-priority queue, IPX is in the normal queue, and FTP (TCP port 21) is in the lowest-priority queue.

Custom Queuing

Custom queuing, or bandwidth allocation, reserves a portion of the bandwidth of a link for each selected traffic type. To configure custom queuing, the network manager must determine how much bandwidth to reserve for each traffic type. If a particular type of traffic is not using the bandwidth reserved for it, then other traffic types may use the unused bandwidth.

Custom queuing works by cycling through the series of queues in round-robin order and sending the portion of allocated bandwidth for each queue before moving to the next queue. If one queue is empty, the router will send packets from the next queue that has packets ready to send. Queuing of packets is still first in, first out in nature in each classification (unless APPN is running in the router, in which case the queue is ordered by SNA transmission priority), but bandwidth sharing can be achieved between the different classes of traffic.



```
Interface Serial 0
ip address 20.0.0.1 255.0.0.0
custom-queue-list 1
!
queue-list 1 protocol ip 1 tcp 2065
queue-list 1 protocol ip 2 tcp 23
queue-list 1 default 3
queue-list 1 queue 1 byte-count 4000
queue-list 1 queue 2 byte-count 2000
queue-list 1 queue 3 byte-count 2000
```

In custom queuing the router is configured to take 4000 bytes from the SNA queue, 2000 bytes from the Telnet queue, and2000 bytes from the IPX queue. This allocates bandwidth in the proportions of 50, 25, and 25 percent. If SNA is not using all its allocated 50 percent of bandwidth, the other queues can utilize this bandwidth until SNA requires it again. The following example shows how to configure custom queuing to allocate bandwidth as illustrated in the picture above.:

Custom queuing is commonly used when deploying DLSw+ networks because it allows the network manager to ensure that a guaranteed percentage of the link can be used for SNA, Telnet, and FTP. However, unless the DLSw+ traffic is broken into separate TCP conversations (using SAP or LOCADDR prioritization described earlier), batch SNA transfer or NetBIOS traffic will share the same output queue and may negatively impact interactive SNA response times.
In Cisco IOS Release 11.0, the number of queues available for custom queuing was increased from 10 to 16. The byte counts you should assign to each queue depend upon the bandwidth of the link and the message sizes of the protocols. Byte counts that are too high may adversely skew the performance of custom queuing on low-speed interfaces.

Considerations

When choosing the byte count values for each queue you must consider the following:

Once the byte count value is exceeded, the frame that is currently being transmitted will be completely sent. Therefore, if you set the byte count to 100 bytes and the frame size of your protocol is 1024 bytes, then every time this queue is serviced, 1024 bytes will be sent, not 100 bytes.

Very large byte counts will produce a "jerky" distribution. That is, if you assign 10,000, 15,000, 20,000, and 25,000 to four queues, each protocol will be serviced nicely when its queue is the one being serviced, but once serviced it may take some time to get back to that queue.

Window size will also affect the bandwidth distribution. If the window size of a particular protocol is set to one, then that protocol will not place another frame in the queue until it receives an acknowledgment. The custom queuing algorithm will move to the next queue if the byte count is exceeded or there are no frames in that queue. Therefore, with a window size of one, only one frame will be sent each time. If your byte count is set to 2 KB and your frame size is 256 bytes, then only 256 bytes will be sent each time this queue is serviced.

You need to know the frame size of each protocol. Some protocols, such as IPX, will negotiate the frame size at session startup time.

Determining the Byte Count

To ensure that the actual bandwidth allocation is as close as possible to the desired bandwidth allocation, you must determine the byte count based on each protocol's frame size. Without doing this, your percentages may not match what you configure.
For example, suppose one protocol has 500-byte frames, another has 300-byte frames, and a third has 100-byte frames. If you want to split the bandwidth evenly across all three protocols, you might chose to specify byte counts of 200, 200, and 200 for each queue. However, that will not result in a 33:33:33 ratio because when the router serviced the first queue, it would send a single 500-byte frame; when it serviced the second

queue, it would send a 300-byte frame; and when it serviced the third queue, it would send two 100-byte frames, giving you an effective ratio of 50:30:20. Had you instead specified 1000, 1000, 1000, the router would send two 500-byte frames, five 200-byte frames, and ten 100-byte frames with a bandwidth ratio of exactly 33:33:33.

However, the delay to send 1000 bytes might be too large. Another alternative is to specify 500, 600, 500, which will result in a ratio of 31:38:31 and may be acceptable.

Fortunately, you do not have to use trial and error to determine the correct byte counts. To determine byte counts, follow these steps:

1.Produce a ratio of all frame sizes, dividing all frame sizes by the largest frame size. For example, assume that the frame size for protocol A was 1086 bytes, for protocol B was 291 bytes, and for protocol C was 831 bytes. The ratios would be:
1086/1086 : 1086/291 : 1086/831

2.Now multiply the results by the percentages of bandwidth you want each protocol to have. In this example we will allocate the following percentages: 20 percent for A, 60 percent for B, and 20 percent for C. This gives us: 1086/1086(0.2) : 1086/291(0.6) : 1086/831(0.2) or .2 : 2.239 : 0.261

3.Again, normalize the ratio by dividing each value by the smallest value, that is: .2/.2 : 2.239/.2 : .261/.2 or 1:11.2:1.3 This is the ratio of the number of frames that must be sent so that the percentage of bandwidth that each protocol uses is approximately in the ratio of 20, 60, and 20 percent.

4.Note that any fraction in any of the ratio values means that an additional frame will be sent. In the example above, the number of frames sent would be one 1086 byte frame, twelve 291-byte frames, and two 831-byte frames, or 1086, 3492, and 1662 bytes, respectively, from each queue. These are the byte counts you would specify in your custom queuing configuration.
To determine the bandwidth distribution this represents, first determine the total number of bytes sent after all three queues are serviced:
(1 x 1086) + (12 x 291) +(2 x 831) = 1086 + 3492 + 1662 = 6240
Then determine the percentage of the 6240 bytes that was sent from each queue:
1086/6240, 3492/6240, 1662/6240 = 17.4, 56, and 26.6 percent

As you can see, this is close to the desired ratio of 20:60:20. The resulting bandwidth allocation can be tailored further by multiplying the original ratio of 1:11.2:1.3 by an integer, and trying to get as close to three integer values as possible. For example, if we multiply the ratio by 2, we get 2:22.4:2.6. We would now send two 1086-byte frames, twenty-three 291-byte frames, and three 831 byte frames, or 2172+6693+2493, for a total of 11358 bytes. The resulting ratio is 19:59:22 percent, which is much closer to the desired ratio than we achieved above.

Do not forget that using a very large byte count may cause other problems.

Weighted Fair Queuing

Weighted fair queuing classifies traffic into conversations and applies priority (or weights) to identified traffic to determine how much bandwidth each conversation is allowed relative to other conversations. Conversations are broken into two categories: those requiring large amounts of bandwidth and those requiring a relatively small amount of bandwidth. The goal is to always have bandwidth available for the small bandwidth conversations and allow the large bandwidth conversations to split the rest proportionally to their weights.

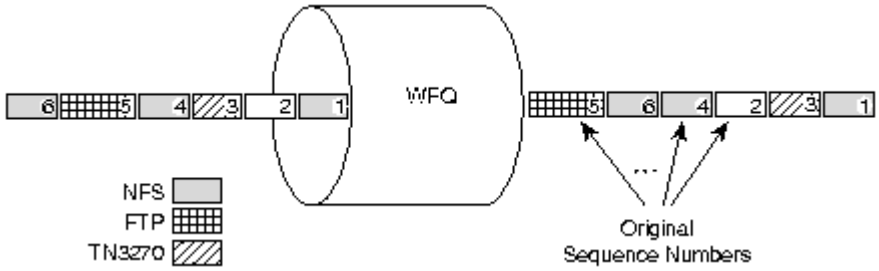
Cisco implements bitwise round-robin fair queuing in Cisco IOS Release 11.0 and later. The prime advantage of fair queuing is that it requires no configuration from the network manager because the router automatically classifies packets passing through an interface into conversations, based on the following:

TCP/User Datagram Protocol (UDP) port address
IP source/destination address, protocol type, type of service
Frame Relay DLCI
X.25 logical channel number (LCN)
SRB frame MAC/SAP

In each case, enough of the packet is checked to break down the streams of packets into separate conversations.

A key disadvantage is that weighted fair queuing does not offer as precise a control over the bandwidth allocation as custom queuing. In addition, in SNA environments, weighted fair queuing typically sees multiple SNA conversations as a single conversation. For example, DLSw+ uses either one or four TCP ports. APPN uses a single LLC2. Hence, instead of SNA interactive sessions moving to the front of the queue, DLSw+ TCP pipes may move to the back of the queue, depending on the number of sessions and quantity of traffic being sent over DLSw+. It is possible, however, to weight certain queues more heavily, which is recommended when using weighted fair queuing in conjunction with DLSw+ or other SNA features. This is covered toward the end of this chapter. In general, do not view weighted fair queuing as an alternative to custom queuing or priority queuing in SNA environments, but simply as a better means of handling default queuing when compared to first in, first out.

In weighted fair queuing, packets between active conversations are reordered so that low-volume conversations are moved forward and high-volume conversations are moved toward the tail of the queue. This reordering results in packet trains being broken up and low-volume conversations receiving preferential service. The high-volume conversations share the delay induced by reordering equally, whereby no one conversation is affected more than another. In Figure 5-7, packets arrive at the router in the order indicated on the left. They are then reordered according to the size and volume of the three conversations so that the packet from conversation 3 (TN3270) is sent second.



Weighted Fair Queuing Reorders Packets on the Output Queue and Packets Within a Single Conversation Are not Reordered. The weighting in weighted fair queuing is currently affected by two mechanisms: IP precedence and Frame Relay discard eligible (DE), forward explicit congestion notification (FECN), and backward explicit congestion notification (BECN). The IP precedence field has values between 0 (the default) and 7. As the precedence value increases, the algorithm allocates more bandwidth to that conversation, which allows it to transmit more frequently.

In a Frame Relay network, the presence of congestion is flagged by the FECN and BECN bits. Once congestion is flagged, the weights used by the algorithm are altered so that the conversation encountering the congestion transmits less frequently.

Switch Products

Supervisor Redundancy
Supervisor resilience is available only on the 13-slot chassis of the Catalyst 5500 switch. No software commands are needed to enable this functionality. You can use the Catalyst 5000 series switch to create a high-speed, fault-tolerant environment that supports mission-critical applications by using a second supervisor module in the chassis of your switch. The second supervisor module takes over in case of the failure of the active supervisor module.

When you use two supervisor modules in the chassis to provide redundancy, the first supervisor module to come up is considered the active module. The second supervisor module remains in standby mode. All network management functions, such as Simple Network Management Protocol (SNMP), command-line interface (CLI) console, Telnet, spanning tree, Cisco Discovery Protocol (CDP), and virtual LAN (VLAN) Trunk Protocol (VTP) take place on the active supervisor module. The Ethernet ports on the standby supervisor module are inactive in the same way that enabled ports on disabled modules are inactive. The console port on the standby supervisor module is also inactive.

The redundant supervisor modules operate in the first two slots of the 13-slot chassis. The supervisor modules are hot-swappable, and in the redundant configuration, the system continues to operate with the same configuration after switching over to the redundant supervisor.

LANE

Communication among LANE components is ordinarily handled by several types of switched virtual channel circuits (VCCs). Some VCCs are unidirectional; others are bidirectional. Some are point-to-point; others are point-to-multipoint.

Control direct VCC--- The LEC, as part of its initialization, sets up a bi-directional point-to-point VCC to the LES for sending or receiving control traffic. The LEC is required to accept control traffic from the LES through this VCC and must maintain the VCC while participating as a member of the emulated LAN.

Control distribute VCC---The LES may optionally set up a unidirectional VCC back to the LEC for distributing control traffic. Whenever an LES cannot resolve an LE_ARP request from an LEC, it forwards the request out the control distribute VCC to all of the clients in the LAN. The control distribute VCC enables information from the LES to be received whenever a new MAC address joins the LAN or whenever the LES cannot resolve an LE_ARP request.

Data direct VCC---Once an ATM address has been resolved by a LEC, this bidirectional point-to-point VCC is set up between clients that want to exchange unicast data traffic. Most client traffic travels via these VCCs.

Multicast send VCC---The LEC sets up a unidirectional point-to-point VCC to a multicast server. This VCC is used by the LEC to sends multicast traffic to the BUS for forwarding out the multicast forward VCC. The LEC also sends out unicast data on this VCC until it resolves the ATM address of a destination.

Multicast forward VCC---The BUS sets up a unidirectional VCC to the LECs for distributing data from the BUS. This can either be a unidirectional point-to-point or unidirectional point-to-multipoint VCC. Data sent by an LEC over the multicast send VCC is forwarded to all LEC's via the multicast forward VCC.

Configure direct VCC---This is a transient VCC which is set up by the LEC to the LECS in order to obtain the LES ATM address which controls a particular LAN that the LEC wishes to join.

Configure the first Lan Emulation Client
Switch> enable

Enter privileged EXEC mode.
Switch#

Enter global configuration mode
Switch# configure terminal

Select the multiservice route processor (CPU) subinterface.
Switch(config)# interface atm 0[.sub_inter#]

Switch(config-if)#

Note We recommend that you configure LECs on subinterfaces (atm 0.1), not main interfaces (atm 0).
Switch(config-if)# lane client-atm-address atm-address-template

Specify an ATM address (and override the automatic ATM address assigned to the LANE client).
Switch(config-if)# lane client {ethernet | tokenring} [elan-name]

Configure a LANE client on the specified subinterface.
Switch(config-if)# end
Switch#