Packet Switching on a Shared Memory Router

Now that we've looked at the general hardware architecture of the shared memory routers and how IOS divides their memory, let's look at how IOS actually switches packets. IOS on shared memory routers supports the following:

- Process switching
- CEF switching
- Fast switching

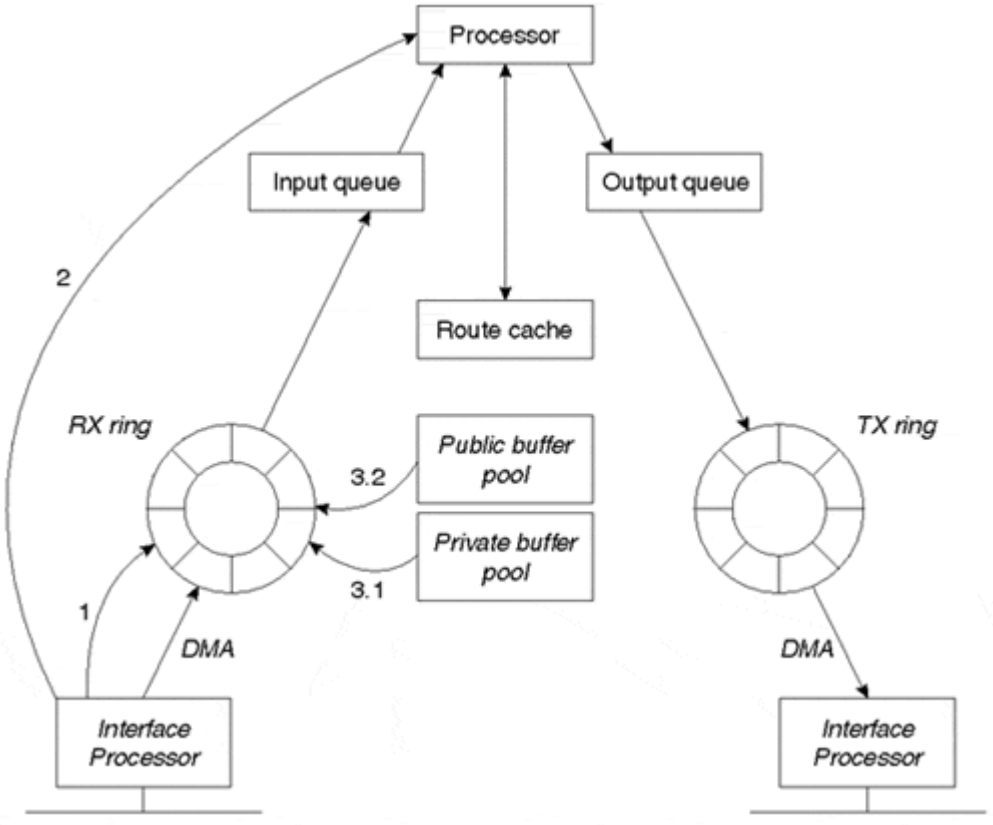
There are three stages in IOS packet switching:

1. Receiving the packet
2. Switching the packet
3. Transmitting the packet

Receiving the Packet

Figure 3-2 illustrates the steps of the packet receive stage.

Figure 3-2. Receiving the Packet



Step 1. The interface media controller detects a packet on the network media and copies it into a buffer pointed to by the first free element in the receive ring (that is, the next buffer the media controller owns). Media controllers use the DMA (direct memory access) method to copy packet data into memory.

Step 2. The media controller changes ownership of the packet buffer back to the processor and issues a receive interrupt to the processor. The media controller does not have to wait for a response from the CPU and continues to receive incoming packets into other buffers linked to the receive ring.

> **NOTE**
>
> Notice in Step 2 the media controller can continue to receive incoming packets into the receive ring. Therefore, it's possible for the media controller to fill the receive ring before the processor processes all the new buffers in the ring. This condition is called an *overrun*; when it occurs, all incoming packets are dropped until the processor catches up.

Step 3. The CPU responds to the receive interrupt, and then attempts to remove the newly-filled buffer from the receive ring and replenish the ring from the interface's private pool. Three outcomes are possible:

Step 3.1. A free buffer is available in the interface's private pool to replenish the receive ring: The free buffer is linked to the receive ring and packet switching continues with Step 4.

Step 3.2. A free buffer is not available in the interface's private pool, so the receive ring is replenished by *falling back* to the global pool that matches the interface's MTU. The *fallback* counter is incremented for the private pool.

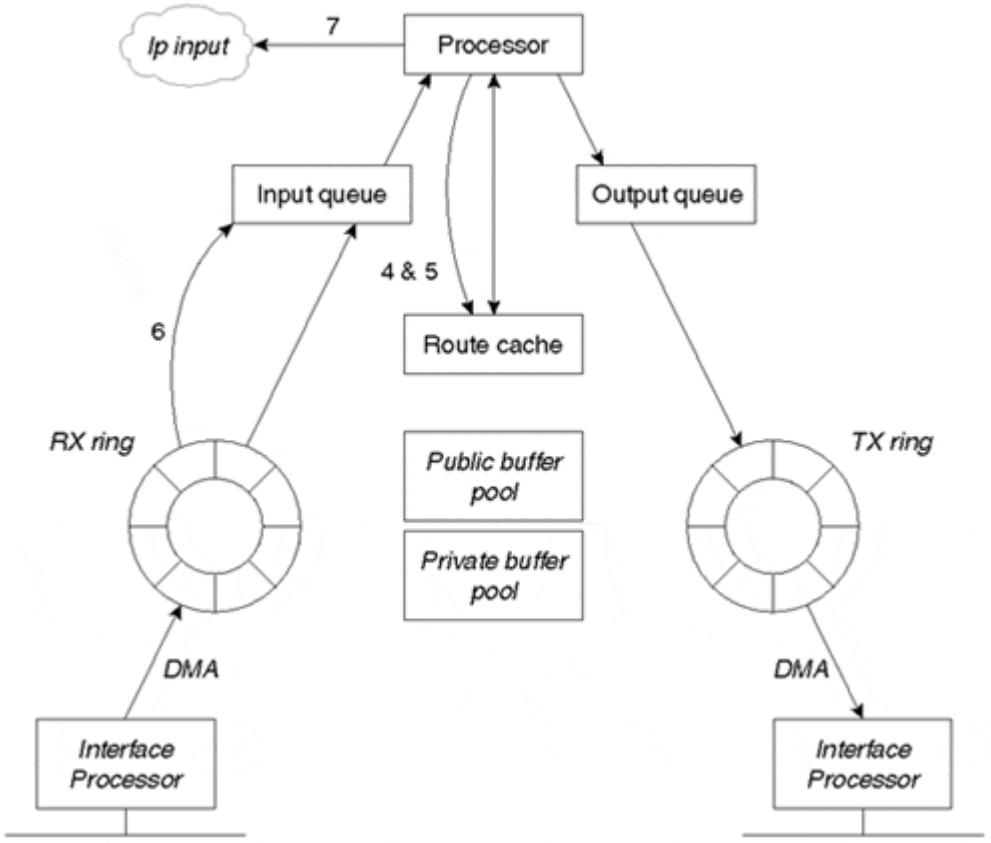
Step 3.3. If a free buffer is not available in the public pool as well, the incoming packet is dropped and the *ignore* counter is incremented. Further, the interface is *throttled*, and all incoming traffic is ignored on this interface for some short period of time.

Switching the Packet

Step 4. After the receive ring is replenished, the CPU begins actually switching the packet. This operation consists of locating information about where to send the packet (next hop) and the MAC header to rewrite onto the packet.

IOS attempts to switch the packet using the fastest method configured on the interface. On shared memory systems, it first tries CEF switching (if configured), then fast switching, and finally falls back to process switching if none of the others work. Figure 3-3 illustrates the steps of the packet switching stage and the list that follows describes the steps illustrated in the figure.

Figure 3-3. Switching the Packet



Step 5. While still in the receive interrupt context, IOS attempts to use the CEF table (first) or fast switching cache (second) to make a switching decision.

Step 5.1.

CEF switching—

If CEF switching is enabled on the interface, then CEF switching is attempted. Four different outcomes are possible:

Step 5.1.1. *If there are valid CEF and adjacency table entries, IOS rewrites the MAC header on the packet and begins transmitting it in Step 8, the packet transmit stage*.

Step 5.1.2. *If the CEF adjacency entry for this destination points to a punt adjacency, IOS proceeds to try fast switching (see Step 5.2)*.

Step 5.1.3. *If the CEF table entry for this packet points to a receive adjacency, the packet is queued for process switching*.

Step 5.1.4. *If there is no CEF entry for the destination, the packet is dropped*.

Step 5.2.

Fast switching—

If CEF is not enabled or the packet cannot be CEF switched, IOS attempts to fast switch the packet.

Step 5.2.1. *If there is a valid fast cache entry for this destination, IOS rewrites the MAC header information and begins transmitting the packet (Step 8, packet transmit stage)*.

Step 5.2.2. *If there is no valid fast cache entry, the packet is queued for process switching (Step 6)*.

Step 6.

Process switching—

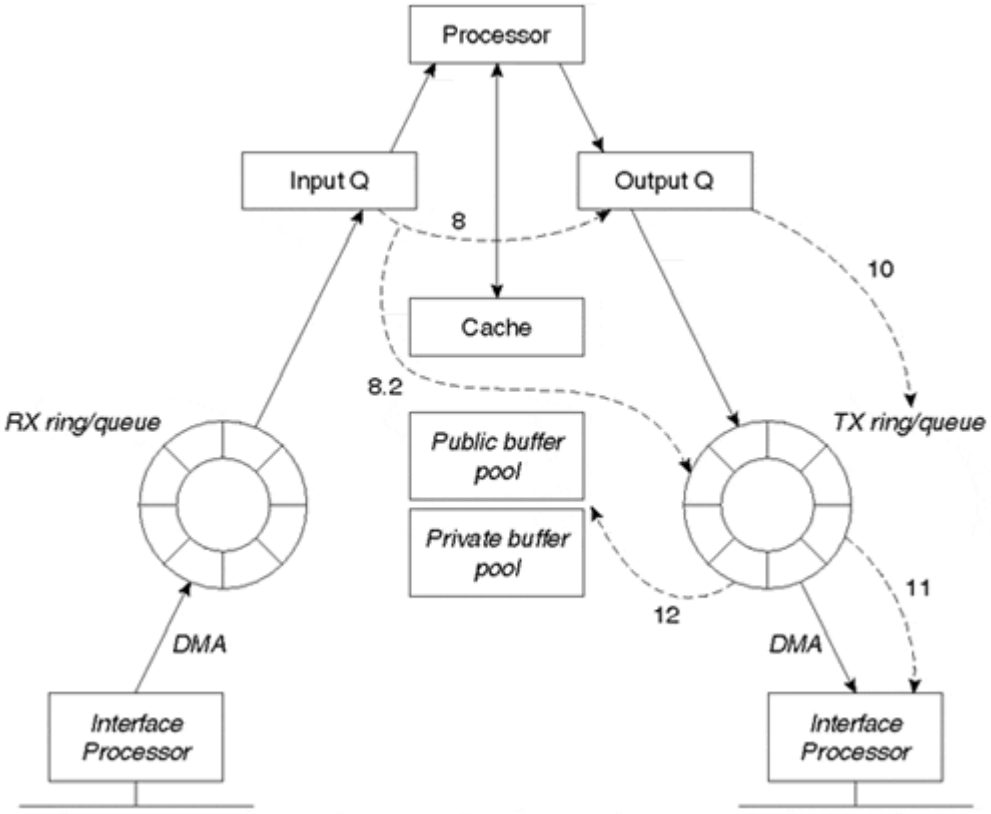
If both CEF switching and fast switching fail, IOS falls back to process switching. The packet is placed in the input queue of the appropriate process (an IP packet is placed in the queue for the IP Input process, for instance), and the receive interrupt is dismissed.

Step 7. Eventually, the packet switching process runs, switching the packet and rewriting the MAC header as needed. Note the packet still has not moved from the buffer it was originally copied into. After the packet is switched, IOS continues to the packet transmit stage, for process switching (Step 9).

Transmitting the Packet

Figure 3-4 illustrates the steps of packet transmit stage.

Figure 3-4. Transmitting the Packet



Step 8. If the packet was CEF switched or fast switched, then while still in receive interrupt context IOS checks to see if there are packets on the output queue of the outbound interface.

Step 8.1. If there are packets already on the output hold queue for the interface, IOS places the packet on the output hold queue instead of directly into the transmit ring to reduce the possibility of out-of-order packets, and then proceeds to Step 8.3.

Step 8.2. If the output hold queue is empty, IOS places the packet on the transmit ring of the output interface by linking the packet buffer to a transmit ring descriptor. The receive interrupt is dismissed and processing continues with Step 11. If there is no room on the transmit ring, the packet is placed on the output hold queue instead and the receive interrupt is dismissed.

Step 8.3. If the output hold queue is full, the packet is dropped, the output **drop** counter is incremented, and the receive interrupt is dismissed.

NOTE

In Step 8 of the transmit stage, notice that IOS checks the output hold queue first before placing any CEF or fast switched packets on the transmit ring. This reduces the possibility of transmitting packets out of order. How can packets get out of order when they're being switched as they're received? Consider the following example.

Let's assume the first packet in a given conversation (flow) between two hosts arrives at the router. IOS attempts to fast switch the packet and finds there is no fast cache entry. So, the packet is handed off to the IP Input process for process switching.

So far, everything is fine. In the process of switching the packet, IP Input builds a cache entry for this flow of packets and places the first packet on the output queue of the outbound interface.

Now, let's assume that just at this moment a second packet arrives for the same flow. IOS fast switches the second packet (because the cache entry has now been built) and gets ready to transmit it. If IOS puts this second packet directly on the transmit ring, it is transmitted before the first packet, which is still waiting in the output queue. To prevent this situation from occurring, IOS always makes certain the output queue of the outbound interface is empty before placing any CEF-switched or fast-switched packets directly on the transmit ring.

Step 9. If the packet was process switched, the packet is placed on the output queue for the output interface. If the output queue is full, the packet is dropped and the *output drop* counter is incremented.

Step 10. IOS attempts to find a free descriptor in the output interface transmit ring. If a free descriptor exists, IOS removes the packet from the output hold queue and links the buffer to the transmit ring. If no free descriptor exists (that is, the ring is full), IOS leaves the packet in the output hold queue until the media controller transmits a packet from the ring and frees a descriptor.

Step 11. The outbound interface media controller polls its transmit ring periodically for packets that need to be transmitted. As soon as the media controller detects a packet, it copies the packet onto the network media and raises a transmit interrupt to the processor.

Step 12. IOS acknowledges the transmit interrupt, unlinks the packet buffer from the transmit ring, and returns the buffer to the pool of buffers from which it originally came. IOS then checks the output hold queue for the interface; if there are any packets waiting in the output hold queue, IOS removes the next one from the queue and links it to the transmit ring. Finally, the transmit interrupt is dismissed.

< BACK Make Note | Bookmark CONTINUE >

Index terms contained in this section