< BACK Make Note | Bookmark CONTINUE >

Packet Switching on the Cisco 7500 Router

IOS on the Cisco 7500 provides the most varied range of switching methods of all the platforms, including:

- Process switching
- Fast switching
- Optimum switching
- Cisco Express Forwarding (CEF)
- Distributed fast switching
- Distributed CEF (dCEF)
- NetFlow switching

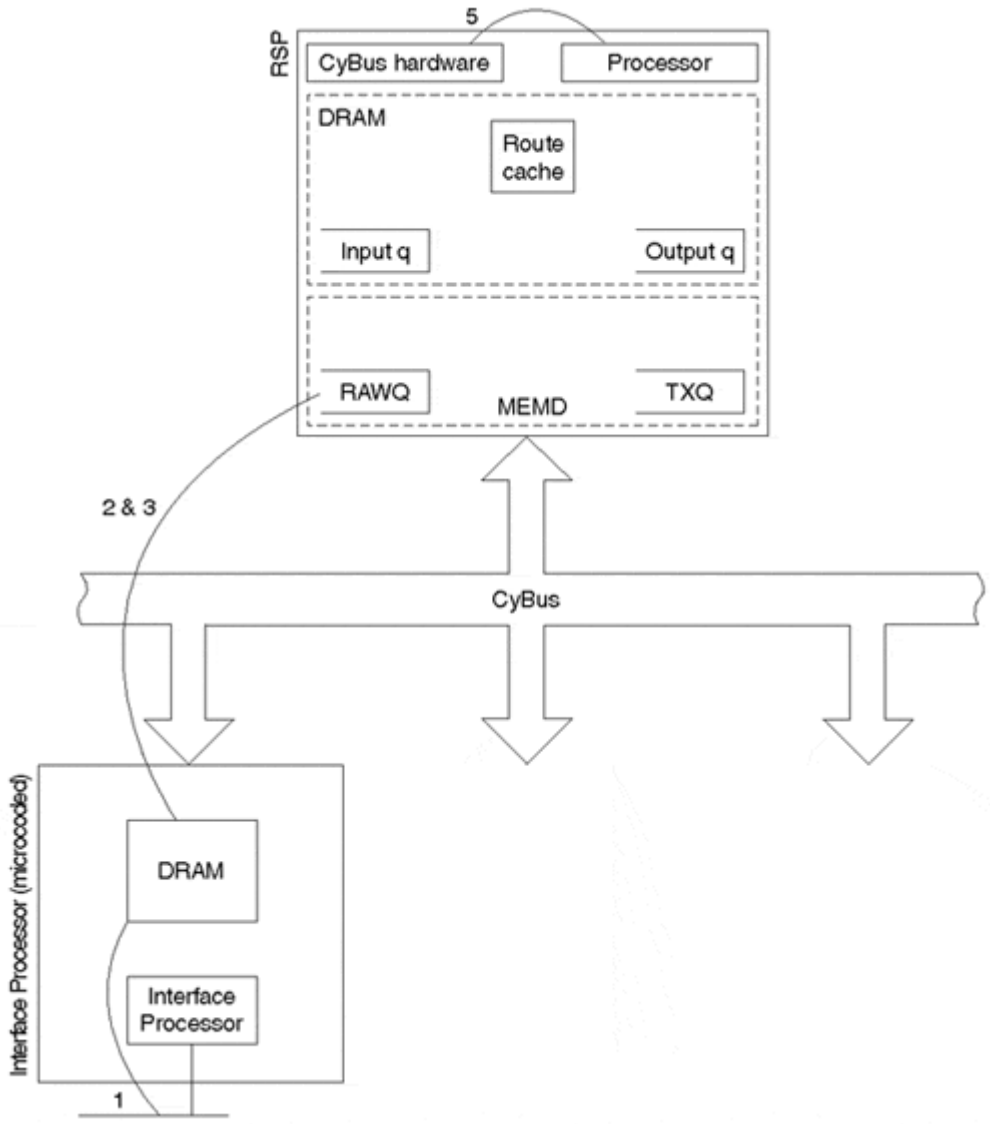
Distributed Fast and Distributed CEF switching are made possible by the VIP interface processor, which is discussed later in this chapter. NetFlow switching is covered in Appendix A.

The following sections describe RSP-based switching. The description is divided into the three stages: receiving the packet, switching the packet, and transmitting the packet.

RSP Switching: Receiving the Packet

Figure 6-4 illustrates the steps of the RSP based packet receive stage; each step is described in the numbered sequence that follows.

Figure 6-4. RSP Switching: Receiving the Packet



Step 1. An interface media controller detects an incoming packet and begins receiving the packet data into the interface processor's internal RAM. Microcode running on the interface processor (for example, an EIP) assembles the data into a contiguous packet in internal RAM.

Step 2. The interface processor microcode attempts to find a MEMD packet buffer for the incoming packet. First, the interface tries to obtain a MEMD buffer from its local free queue (lfreeQ). If there are no free buffers in the local free queue, the microcode attempts to obtain a MEMD buffer from the interface's assigned global free queue (gfreeQ). There are three possible outcomes:

Step 2.1. There might be no free MEMD buffers in the global pool, in which case the packet is dropped and the interface's **ignore** counter is incremented.

Step 2.2. Free buffers are available in the global pool, but the interface can't obtain any because it's already holding the maximum buffers allowed by its *rxhi* value (that is, rxcurr = rxhi). Again, in this case, the packet is dropped and the interface's **ignore** counter is incremented.

Step 2.3. The microcode successfully obtains a free buffer from the global free queue.

Step 3. After a free MEMD packet buffer is obtained, the microcode copies the packet data from its internal RAM across the CyBus to the MEMD buffer.

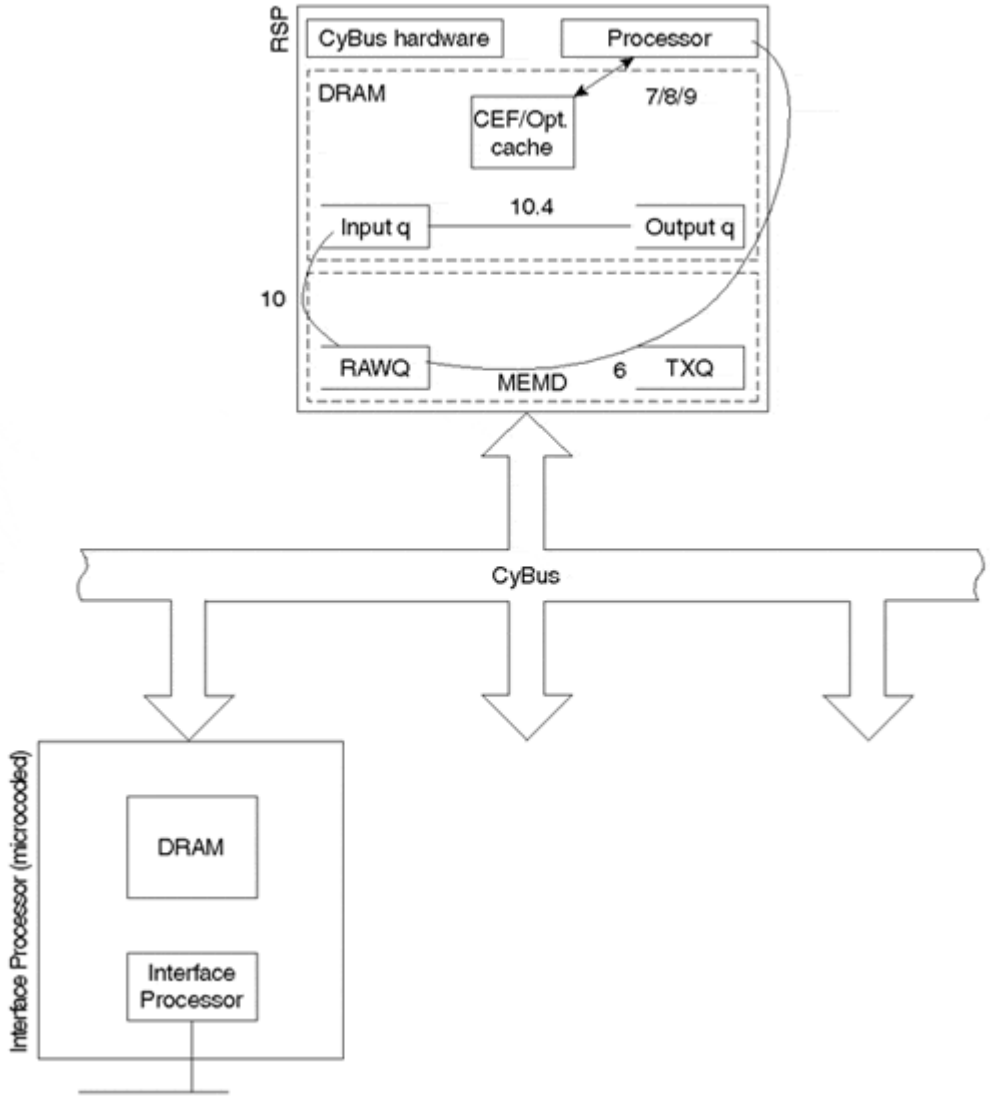
Step 4. The interface processor then enqueues the MEMD buffer header onto the Raw Queue (RawQ) for processing by the RSP.

Step 5. Special CyBus hardware detects a buffer has been placed in the RawQ and raises a network interrupt to the RSP CPU.

RSP Switching: Switching the Packet

Figure 6-5 illustrates packet switching on the RSP. The numbered list following describes each step.

Figure 6-5. RSP Switching: Switching the Packet



Step 6. The RSP CPU acknowledges the network interrupt. IOS begins processing the packet by removing the MEMD buffer header from the RawQ, locating the MEMD packet buffer, and inspecting the buffer's contents. Note the remainder of these steps assume IOS is processing only one packet during a network interrupt. In practice, IOS continues to process packets during an interrupt as long as there are buffer headers waiting on the RawQ. IOS now decides how the packet should be switched.

Step 6.1. If the contents are an IP packet and the inbound interface is configured for CEF switching, go to CEF switching.

Step 6.2. If the contents are an IP packet and the inbound interface is configured for optimum switching (on by default on most interfaces), then go to optimum switching. If the input interface is not configured for optimum switching, then go to fast switching.

Step 6.3. If the contents are an IP packet and the inbound interface is configured for NetFlow switching,

then go to NetFlow switching.

Step 6.4. If the contents are not an IP packet, then go to fast switching by default.

Step 7. CEF switching. While still processing the network interrupt, the CPU looks through the CEF cache for the outbound interface and next hop.

Step 7.1. There is a valid entry in the CEF table for the destination. The packet is re-encapsulated in the same MEMD buffer, and processing continues in the transmit stage.

Step 7.2. The packet is destined to the router itself or is a control packet, such as a routing update. In this case, the packet is prepared for process switching (see "Process Switching" in Step 10).

Step 7.3. There is no matching CEF entry. In this case, the packet is dropped, the CEF drop counter is incremented, the CPU completes its interrupt processing, and the network inter-rupt is dismissed (this step is not illustrated in Figure 6-5).

Step 7.4. The CEF lookup leads to a punt adjacency. The packet is passed to fast switching. This situation occurs, for example, if the outbound interface encapsulation is not supported by CEF.

Step 8. Optimum switching. While still processing the network interrupt, the CPU looks through the optimum cache to determine which interface and next hop information to use in switching the packet.

Step 8.1. If the forwarding information is found in the optimum cache, the packet is re-encapsulated in the same MEMD buffer and processing continues in the transmit stage.

Step 8.2. If there is no entry in the optimum cache, the packet is passed to fast switching.

Step 9. Fast switching. While still processing the network interrupt, the CPU does a lookup of the destination IP address in the fast cache.

Step 9.1. If the cache lookup is successful, the packet is re-encapsulated in the same MEMD buffer and processing continues in the transmit stage.

Step 9.2. If the cache lookup is not successful or the packet is destined for the router itself, the packet is prepared for process switching (see Step 10).

Step 10. Process switching.

Step 10.1. While still processing the network interrupt, the CPU looks for a free system buffer in which to copy the packet from the MEMD buffer. Process switched method packets must be copied into a system buffer in main memory because MEMD packet buffers are reserved for interrupt level packet processing only (to prevent MEMD buffer starvation).

Step 10.1.1. *If IOS fails to get a system buffer, the* ***input drop*** *and the* ***no buffer*** *interface counters are incremented and the packet is dropped. The CPU finishes processing the interrupt and it's dismissed.*

Step 10.1.2. *If a system buffer is available but the interface's input hold queue count is at its maximum, the interface is throttled, the* ***input drop*** *counter is incremented, and the packet is dropped. IOS finishes processing the interrupt and dismisses it. By default, interfaces are allowed to have 75 packets in input hold queues waiting to be processed.*

Step 10.1.3. *If the CPU succeeds in obtaining a system buffer, the packet is copied from MEMD to the system buffer and the MEMD buffer is returned to the local free queue of the interface. The* ***input hold queue*** *count is incremented.*

Step 10.2. The packet (now contained in a system buffer) is enqueued to a switching process based on its type. The CPU finishes processing the interrupt and dismisses it.

Step 10.3. The packet is eventually switched by a background process; see Chapter 2, "Packet Switching Architecture" for more details.

Step 10.4. After the packet has been switched, it's placed in the outbound interface's output hold queue. If the output hold queue is full (the default maximum is 40 packets), the packet is dropped and the outbound interface's **output drop** counter is incremented. If the packet is dropped, the inbound interface's **input hold queue** count will also be decremented at this point.

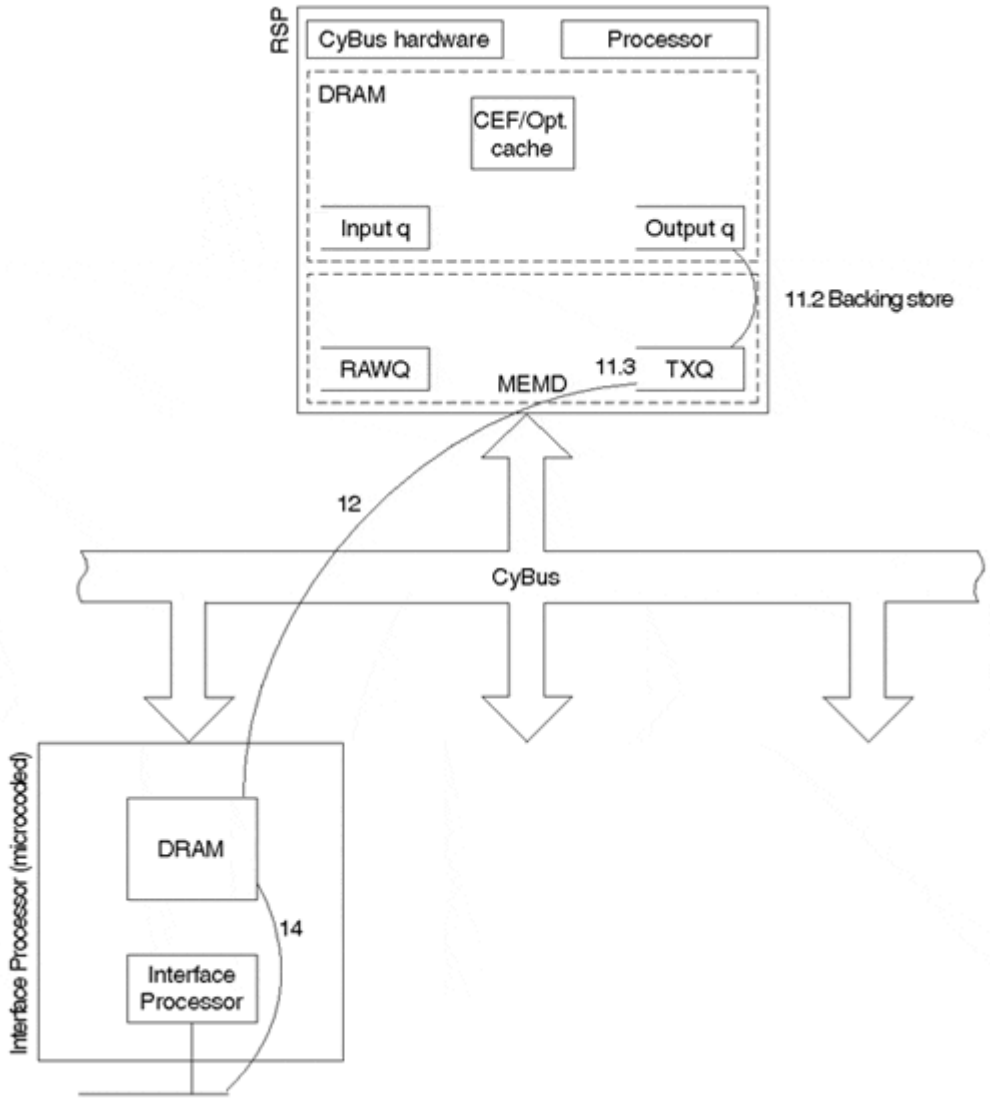
Step 10.5. IOS attempts to obtain a MEMD packet buffer so the packet can be transmitted by the interface processor. IOS attempts to obtain the MEMD buffer from the outbound interface's global free list. If there are no available MEMD buffers, or the outbound interface's transmit queue is full, IOS leaves the packet on the output hold queue (in a system buffer) and tries again later to output the packet.

Step 10.6. IOS copies the packet from the system buffer into the MEMD packet buffer. The system buffer is removed from the output hold queue, freed, and returned to the system buffer pool. The input hold queue count for the inbound interface is also decremented at this point. IOS then moves to the packet transmit stage.

RSP Switching: Transmitting the Packet

Figure 6-6 illustrates the steps of the packet transmit stage for packet switching. The numbered sequence that follows describes each step.

Figure 6-6. RSP Switching: Transmitting the Packet



Step 11. IOS attempts to place the MEMD buffer header onto the transmit queue of the outbound interface.

Step 11.1. If the outbound interface's transmit queue is full and the interface is not configured for *backing store*, the packet is dropped and the **output drop** counter is incremented.

Step 11.2. If the outbound interface's transmit queue is full and the interface *is* configured for backing store, the packet is copied from the MEMD buffer to a system buffer and the system buffer is placed on the interface's output hold queue. The **output buffer's swapped out** counter is incremented. Note that if the outbound interface is configured for any type of fancy queueing (such as weighted fair queueing), backing store is enabled by default and cannot be disabled.

Step 11.3. If the outbound interface's transmit queue is not full, the MEMD buffer header is placed on the queue and the value of the transmit accumulator (that is, the number of additional buffer headers that can be added to the queue) is decremented.

Step 11.4. If IOS is still processing within a network interrupt, the CPU completes the interrupt processing and it is dismissed.

Step 12. The microcode on the outbound interface processor detects the buffer header in the transmit queue and copies the contents of the MEMD packet buffer across the CyBus into its internal RAM.

Step 13. The MEMD buffer header and buffer are freed and returned to the *inbound* interface's local free queue (lfreeq).

Step 14. The outbound interface media controller transmits the packet from the interface processor's internal RAM out to the media.

> **NOTE**
>
> Backing store is designed to prevent the router from dropping packets during bursts. Backing store causes packets that have already been switched to be copied to the outbound interface's output queue when the txQ for the interface is full—an extremely expensive operation. This feature should never be enabled on any interface with a bandwidth greater than 2 MBps because it can cause extreme stress on the router's CPU.

< BACK Make Note | Bookmark CONTINUE >

Index terms contained in this section