

Part I: Understanding, Configuring, and Troubleshooting CEF

Chapter 1 Introduction to Packet Switching Architectures

Chapter 2 Cisco Express Forwarding

Chapter 3 CEF Enhanced Scalability

Chapter 4 Basic IP Connectivity and CEF Troubleshooting

Chapter 1. Introduction to Packet-Switching Architectures

This chapter covers the following topics:

- Routing and switching
- Understanding router pieces and parts
- Cisco IOS Software: the brains
- Processes and scheduling
- Putting the pieces together: switching a packet
- Hardware and software show commands

What, exactly, does a router do? What's the difference between a switch and a router? If you were building a router, what would the parts of the router be? How would they fit together? These are the sorts of questions that router designers within Cisco face every day, both from the hardware and software points of view.

This chapter begins with a discussion of the terms routing and switching and provides you with the background needed to understand the differences between the two. The chapter then covers the physical pieces and parts of a router and discusses the brains, Cisco IOS Software. You then learn how the pieces work together to switch a packet.

Routing and Switching

The networking industry uses many terms and concepts to describe switching and routing; because a good number of them have overlapping meanings, deciphering the terminology can be confusing. Does a router route or switch? What's the difference between Layer 3 switching and routing? What's Layer 7 switching, and who cares? Let's examine what happens to a packet as it passes through a network to try and discover some of the answers to these questions.

Understanding Broadcast and Collision Domains

The two primary concepts you need to understand when discussing the various concepts of switching are broadcast domain and collision domain. The simple network in [Figure 1-1](#) illustrates these two concepts.

Figure 1-1. Broadcast and Collision Domains



The collision domain is defined as the set of hosts that are attached to the network which might not transmit at the same time without their transmissions colliding. For example, if Host A and Host B are connected through a straight wire, they cannot transmit at the same time. If, however, some physical device between them allows them to transmit at the same time, they are in separate collision domains.

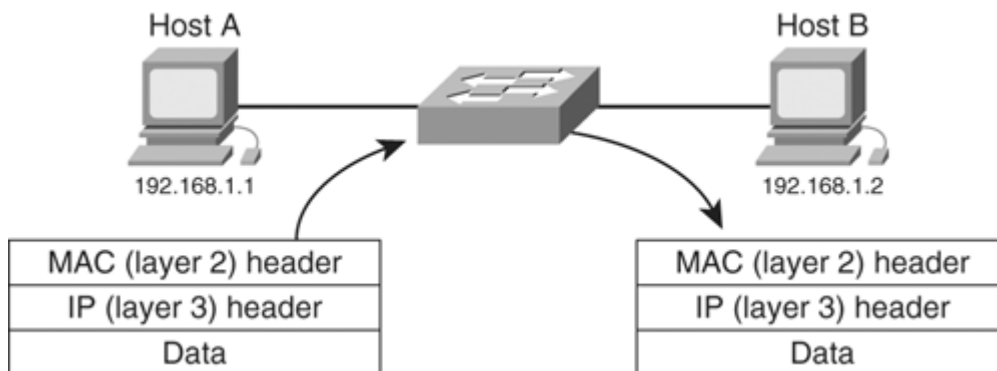
The broadcast domain is the set of hosts that can communicate simply by sending Layer 2 (or link-layer) broadcasts. If Host A transmits a broadcast packet to all the hosts that are locally attached, and Host B receives it, these two hosts are in the same broadcast domain.

Broadcast and Collision Domains

Bridging breaks up the collision domain, but not the broadcast domain. In fact, traditional switching and bridging are the same thing technically. The primary difference is that in most switched environments, each device connected to the network is in a separate collision domain.

Looking at the format of a typical data packet, what is changed when the packet crosses a switch? Not a single thing, as Figure 1-2 illustrates.

Figure 1-2. Packets Passing Through a Switch



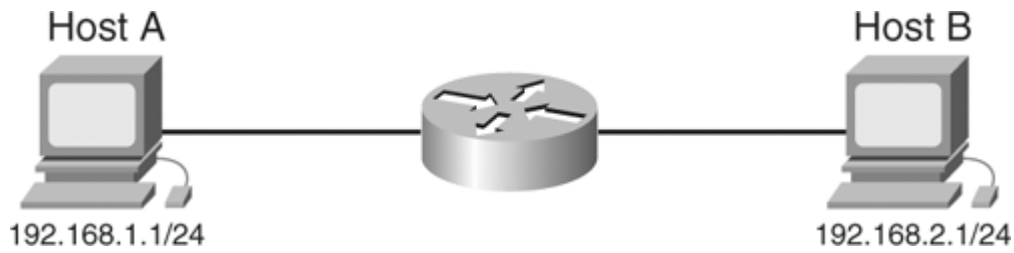
In effect, devices on either side of a switch cannot tell that there is a switch between them, nor do they know the destination of their packets; switches are transparent to devices connected to the network.

If Host A wants to send a packet to 192.168.1.2 (Host B), it can send a broadcast to all the hosts connected to the same segment asking for the MAC address of the host with the IP address 192.168.1.2 (this is called an Address Resolution Protocol (ARP) request). Because Host B is in the same broadcast domain as Host A, Host A can be certain that Host B will receive this broadcast and answer with the correct MAC address to send packets to.

Broadcast and Collision Domains in Routing

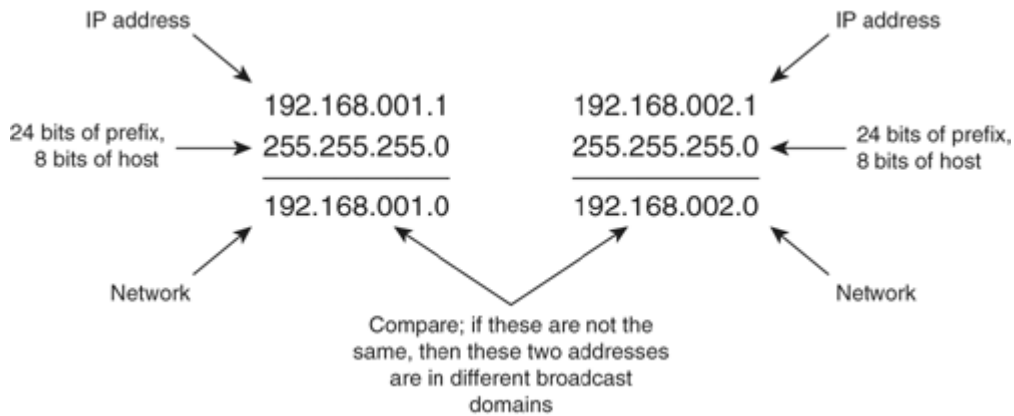
Routers not only break the collision domain, but they also break the broadcast domain, as Figure 1-3 illustrates.

Figure 1-3. Routing



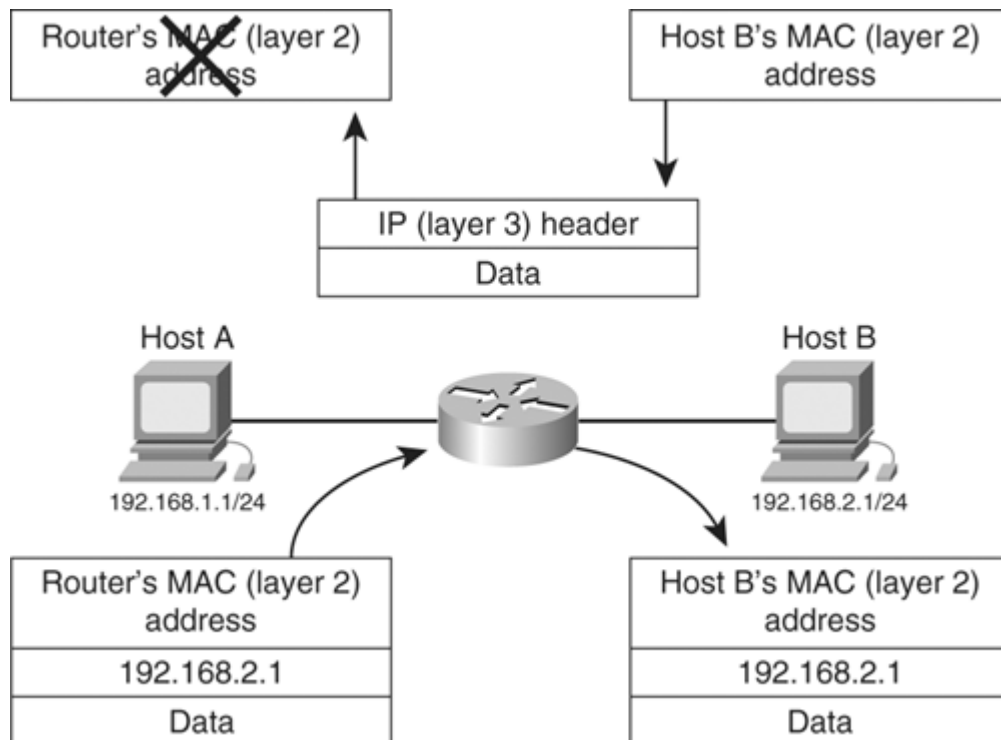
Now, if Host A wants to reach 192.168.2.1, how will it do this? It cannot broadcast an address resolution packet to discover Host B's address, so it has to use some other method to figure out how to reach this destination. How does Host A know this? Note that after each IP address in Figure 1-3, there is also a /24; this number indicates the prefix length, or the number of bits that are set in the subnet mask. Host A can use this information to determine that Host B is not in the same broadcast domain (not on the same segment), and Host A must use an intervening router to reach the destination, as Figure 1-4 shows.

Figure 1-4. Determining That a Destination Is Not in the Same Broadcast Domain
[View full size image]



Now that Host A knows Host B isn't in the same broadcast domain, it also knows it can't send a broadcast out to resolve Host B's address. How, then, can Host A reach Host B? By directing its packets toward the intervening router. Host A places Host B's IP address in the packet header, but it places the intervening router's MAC address in the packet, as Figure 1-5 shows.

Figure 1-5. Packet Flow Through a Router

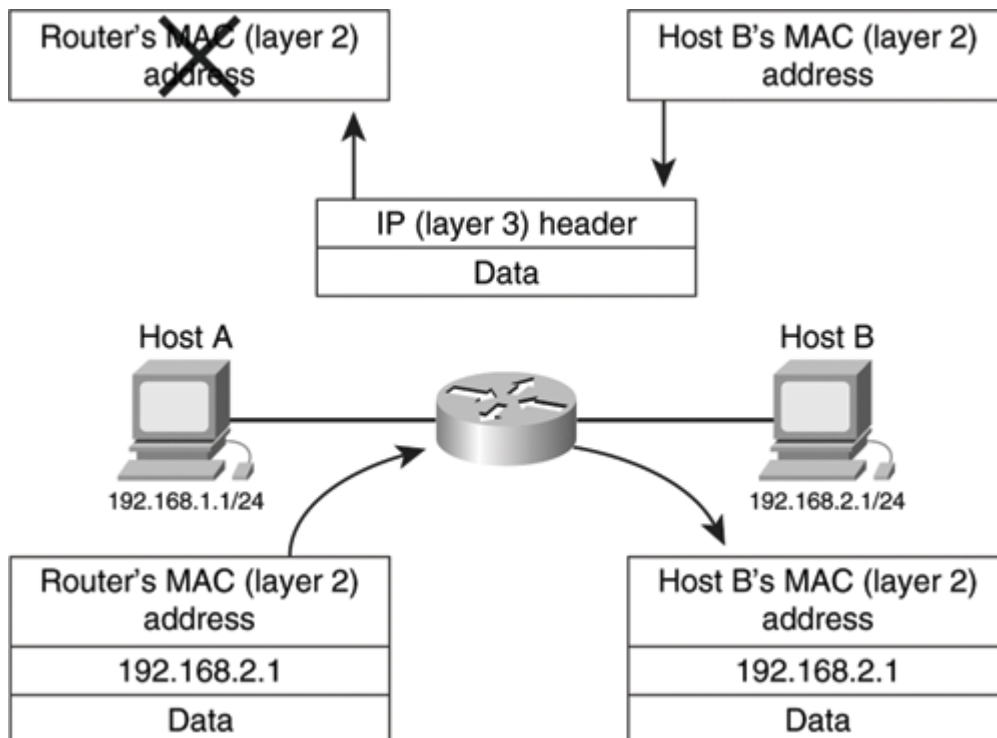


Host A puts the router's MAC address on the packet, so the router accepts the packet off the network. The router examines the destination IP address and determines what the next closer hop should be by consulting a routing table (in this case, it is Host B itself), and replaces the MAC address with the correct MAC address for the next hop. The router then transmits the packet back onto a different segment, which is in a different broadcast domain.

Layer 3 Switching

Layer 3 switching looks a lot like routing, as [Figure 1-6](#) illustrates (note that it is the same as [Figure 1-5](#)). That's because Layer 3 switching is routing, as far as hosts connected to the network are concerned; there is no functional difference between Layer 3 switching and routing.

Figure 1-6. Layer 3 Switching



Note

The rest of this book refers to the process of forwarding a packet based on Layer 3 information from a cache or table switching, without referencing Layer 3. The rest of the book provides little or no discussion of Layer 2, or traditional, switching; in those places where Layer 2 switching is involved, it will be explicitly stated.

Understanding Router Pieces and Parts

Consider the following components required to switch a packet:

- Some device detects packets that the router needs to receive (pick up off the wire) and process.
- The packet needs to be stored (copied someplace in the router's memory).
- The packet's header must be examined (by the local packet psychologist to see whether the packet is crazy), various options and fields within the packet (such as the Time to Live field) must be processed, and the Layer 2 header needs to be replaced with the correct Layer 2 header for the next hop of the packet's journey.
- The packet must then be given to some device on the outbound interface that will transmit it onto the correct network segment.

Each of these four steps must be performed by some piece of hardware or software within the router.

Most routers use the following fundamental pieces of hardware while switching packets:

- Interface processors
- Central processing unit (CPU)
- Memory
- Backplane and switching fabric

The following sections discuss each of these pieces.

Interface Processors

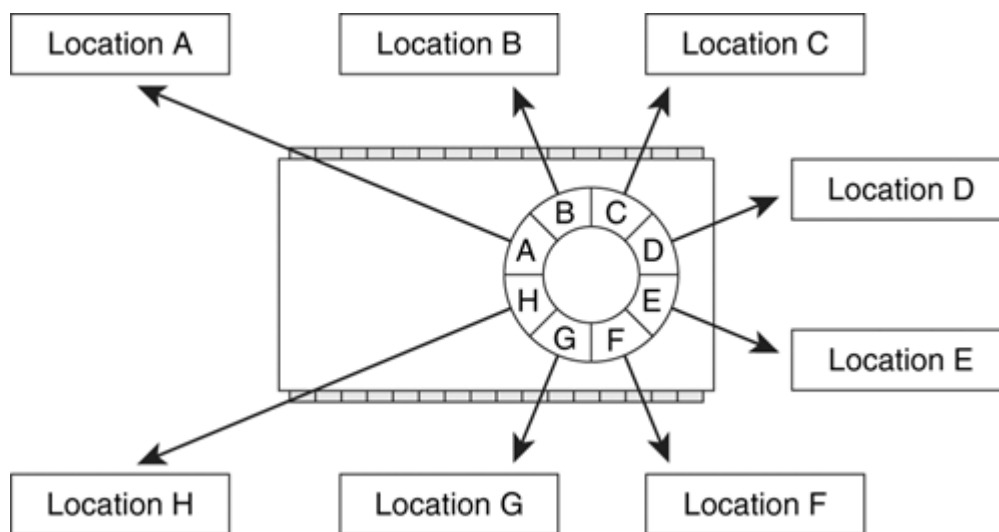
Interface processors are responsible for the following actions:

- Decoding the electrical or optical signals as they arrive from the physical media
- Translating these signals into 1s and 0s
- Transferring the 1s and 0s into a memory location
- Deciding where the end of a packet is and signaling other devices in the router that there is a new packet to process

Typically, interface processors are commercially available chips designed specifically to decode a particular type of signal and translate it into packets. For example, the older Lance Ethernet chipset is a well-known decoder for Ethernet networks that translates the Ethernet electrical encoding into packets.

Interface processors transfer packets into memory using direct memory access; they will directly copy the packet into a memory location specified by the controlling device (in this case, a Cisco IOS Software device driver, which we cover in the section "[Cisco IOS Software: The Brains](#)," later in this chapter). The set of addresses they copy the packets into is stored as a ring buffer, which is illustrated in [Figure 1-7](#).

Figure 1-7. Interface Processor Ring Buffers



Each entry in the buffer points to a different memory location; the first packet copied in off the wire will be placed in the memory location indicated by (pointed to by) A, while the second will be placed in the memory location pointed to by B, the third will be placed in the memory location pointed to by C, and so on.

When the interface processor copies a packet into location H, it will loop around the ring and start copying the next packet into location A again. This looping effect is why the transmit and receive buffers are generally called transmit and receive rings.

Central Processing Unit

The central processing unit (CPU) provides horsepower for any general task the software needs to perform. On some platforms, the CPU does the work required to switch packets, whereas on others, the CPU primarily focuses on control-plane management while hardware specifically tailored to switching packets does the packet switching.

Memory

Cisco routers use memory to store the following:

- Packets while they are being switched
- Packets while they are being processed
- Routing and switching tables
- General data structures, executing code, and so on

Some Cisco platforms have only one type of memory, dynamic random-access memory (DRAM) or synchronous dynamic random-access memory (SDRAM), whereas others have a wide variety of memory available for different purposes. You learn how memory is divided up and used in the section "[Memory Management](#)," later in this chapter.

Backplanes and Switching Fabrics

When a packet is switched by a router, it has to be copied from the inbound port to the outbound port in some way; the ports in the router are generally interconnected using some sort of a switching fabric to allow this inbound-to-outbound copying. Cisco routers use three types of interconnections:

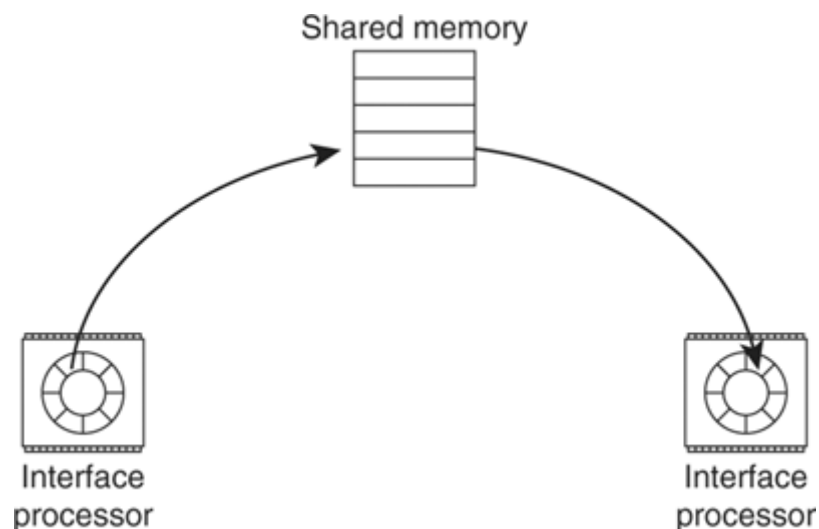
- Shared memory
- Crossbar switching fabric
- Bus backplanes

The following sections describe each type.

Shared Memory

In shared memory architectures, packets are copied into a memory location that is accessible by both the input and output interface processors. [Figure 1-8](#) illustrates that as a packet is copied into memory on the inbound side, it is copied into a packet buffer that all interfaces can access. To transmit a packet, the outbound interface copies the packet out of the shared memory location onto the physical wire (encoding the 0s and 1s as needed).

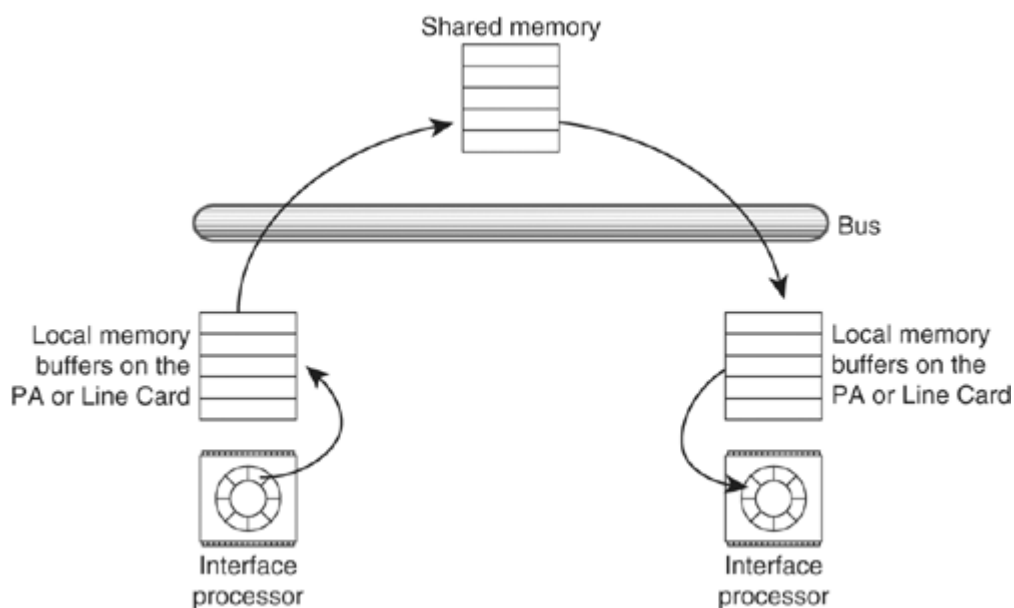
Figure 1-8. Copying Packets Through Shared Memory



The primary speed bottleneck on shared memory architectures tends to be the speed of the memory in which the packets are stored. If the memory cannot be accessed for some amount of time after a packet is written to it, or if the access speed is slow, the interface processors won't be able to copy packets to and from the wire quickly. Most routers that use this architecture use very-high-speed memory systems to accommodate high-speed packet switching.

In some routers, the interface processor is physically separated from the shared memory used to switch packets; for example, in the Cisco 7200 series of routers, the interface processors are physically located on the Port Adapters (PAs), while the shared memory is physically located on the Network Processing Engine (NPE). In these cases, a bus separates the shared memory from the interface processor, as [Figure 1-9](#) illustrates.

Figure 1-9. Copying Packets Through Shared Memory over a Bus
[\[View full size image\]](#)



If there is a bus between the interface processor and the shared memory, packets are copied into local memory on the line card and then transferred across the bus to the shared memory, where other line cards can access the packet to copy it back for transmitting.

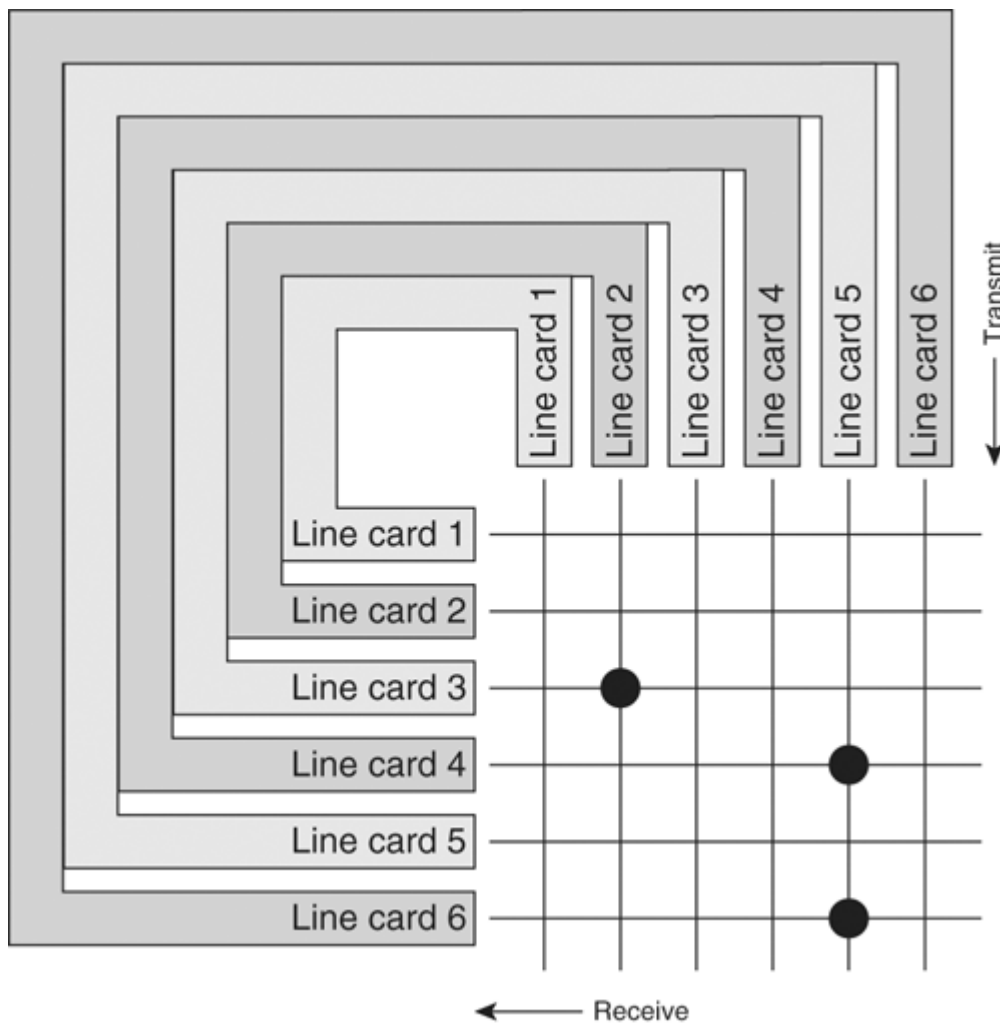
Because each packet must be copied across the bus between the interface processor and the shared memory twice, the bus's bandwidth has a major impact on the performance of the router, as well as on the speed at which packets can be written into and read from the shared memory.

In virtually all shared memory systems, whether there is a bus between the interface processor and the shared memory or not, the work of switching the packet is done by the CPU or some other centralized processor that has access to the shared memory.

Crossbar Switching Fabric

If a switching decision can be made on individual line cards—such as when the line cards have specialized hardware or a separate processor—copying the packet into shared memory of any type between interfaces isn't necessary. Instead, the packet can be transferred directly from the local memory on one line card to the local memory on another line card. [Figure 1-10](#) illustrates an example of a crossbar switching fabric.

Figure 1-10. Crossbar Switching Fabric



In a crossbar switching fabric, each line card has two connections to a fabric. At each cycle (or point in time), any given line card's output can be connected to any other line card's input. So, if Line Card 2 receives a packet that it determines needs to be transmitted out a port on Line Card 3, it can ask the switch fabric controller to connect its output to the input of Line Card 3 and transfer the packet directly.

For multicast, the inbound line card can request a connection to multiple line card inputs. For example, in [Figure 1-10](#), Line Card 5's output is connected to both Line Card 6's and Line Card 4's input, so any packets Line Card 5 transmits will be received by both of these line cards at the same time.

The primary advantage of a crossbar switching fabric is that it can be scaled along with the number of line cards installed in a system; each new line card installed represents a new set of input and output connections that can be used in parallel with the other existing connections. As long as each individual connection has enough bandwidth to carry line-rate traffic between any pair of line cards, the entire router can carry line-rate traffic between multiple pairs of line cards.

The bandwidth of the individual lines in the crossbar switching mesh will not help in one instance: if two line cards want to transmit packets to a third line card at the same time. For example, in [Figure 1-10](#), if both Line Cards 1 and 2 want to transmit a packet to Line Card 3, they can't; Line Card 3 has only one input.

This problem has several solutions, any and all of which are used in Cisco routers. The first is to schedule the connections in a way that no line card is expected to receive from two other line cards at once. Cisco routers

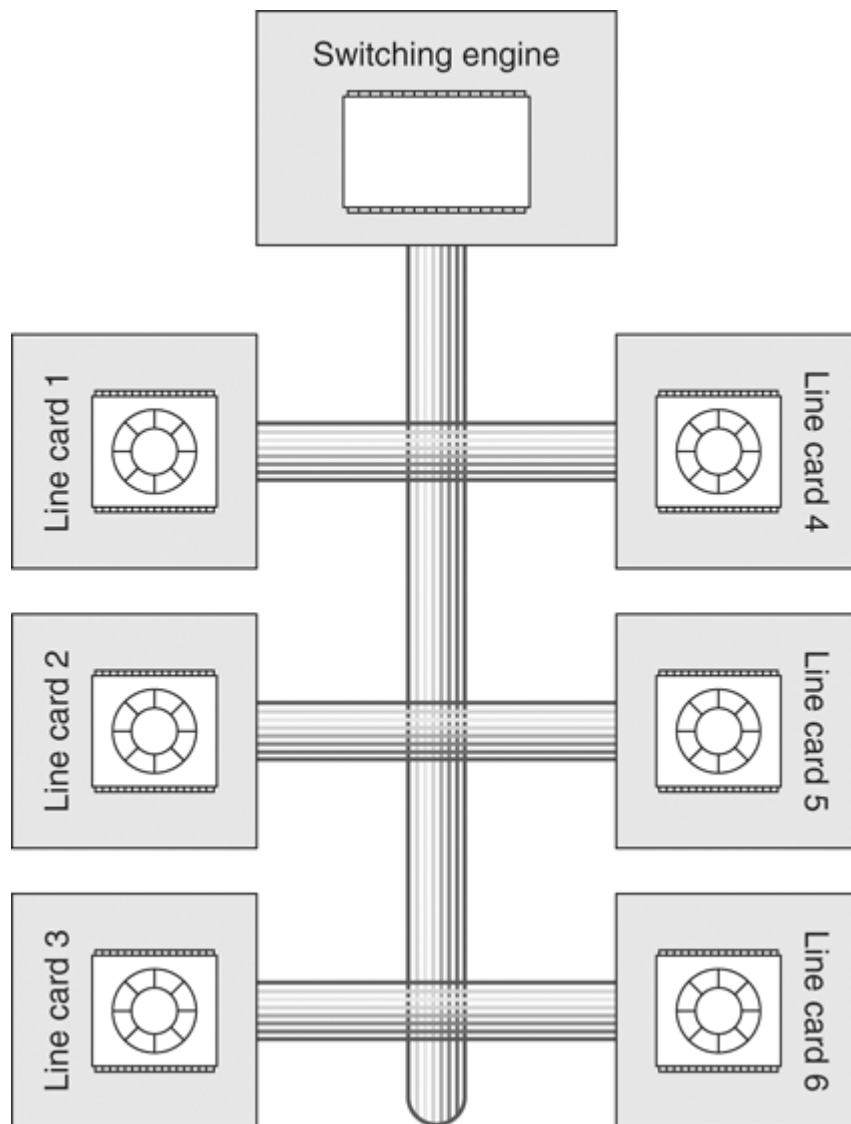
provide various scheduling algorithms and systems to ensure that no line card is starved for transmission, so packets don't bottleneck on the inbound line card.

Another possibility, used on some high-speed line cards on some platforms, is to provide two connections into a given line card so that it can receive packets from two different line cards at the same time. This doesn't completely resolve the problem—scheduling is still required—but it does relieve some of the pressure on line cards that are often used as uplinks to higher-speed networks.

Bus Backplanes

The last type of connection used between line cards is a bus backplane. This is different in a few ways from a shared memory architecture with a bus. [Figure 1-11](#) illustrates such a backplane.

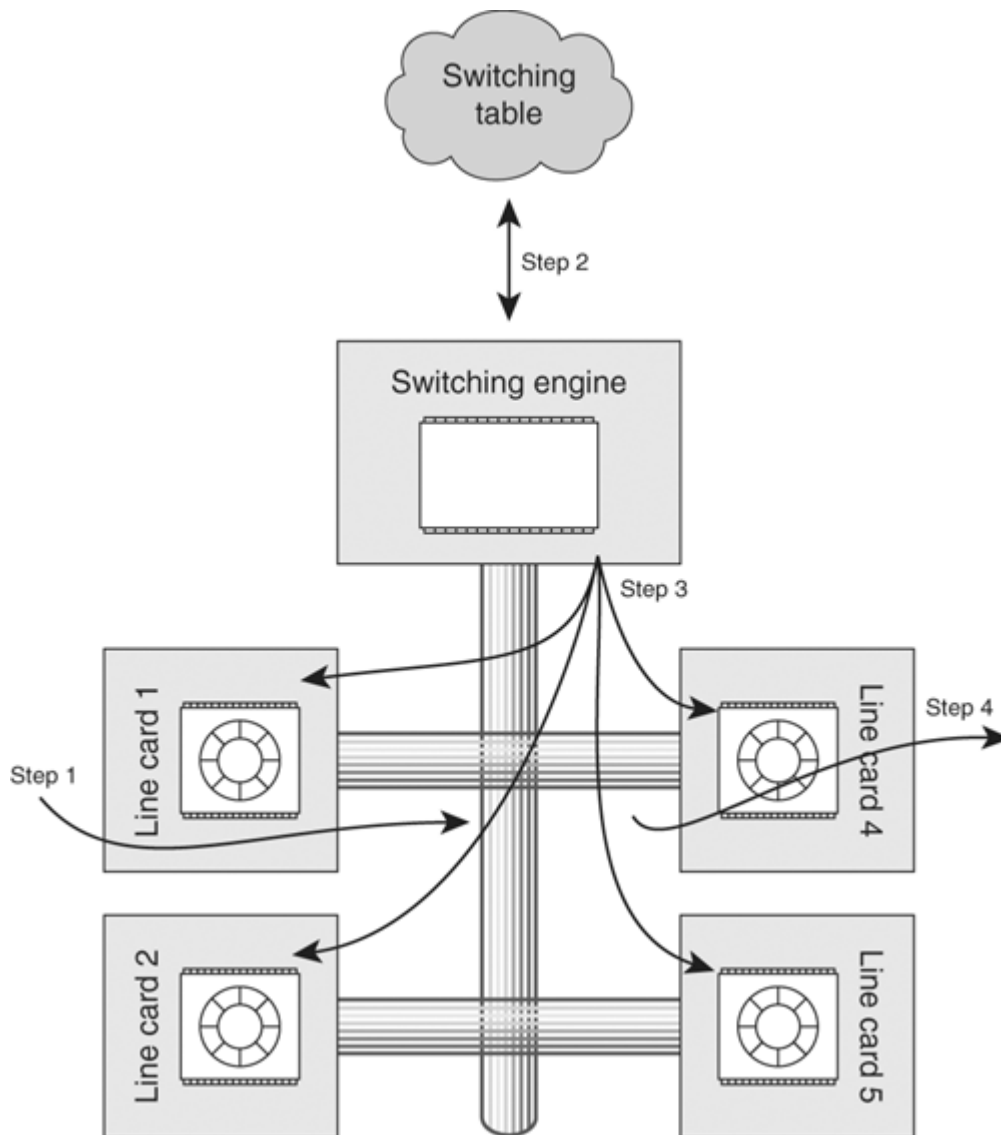
Figure 1-11. Bus Backplane



Packets that are received by a line card are transmitted onto the backplane as they are received. When the switching engine has received and processed enough of the packet to decide which interface the packet needs to be transmitted out of, the line card with that interface connected is instructed to continue copying the packet off the backplane and is given information about which port the packet should be transmitted out.

The remainder of the line cards is instructed to flush this packet from their buffers and prepare to receive the next packet, or do other work as needed. Figure 1-12 illustrates how this works.

Figure 1-12. Packet Receipt and Transmission Across a Bus Backplane



The following list explains the process illustrated in Figure 1-12:

1. Line Card 1 begins receiving the packet and copies it onto the backplane; all the other line cards begin to store the packet in local buffers.
2. When Line Card 1 has put the packet headers up the IP header onto the backplane, the switching engine consults its local tables to determine the correct output port.
3. The switching engine now signals Line Card 4 that it should continue accepting and storing this packet, and it informs the remaining line cards that there is no reason for them to continue receiving or storing this packet.
4. After Line Card 1 has finished transmitting the packet onto the backplane, Line Card 4 transmits the packet onto the locally connected physical media.

Switching multicast packets over a backplane bus is relatively simple; the switching engine signals several line cards to accept and transmit the packet, rather than just one. Scheduling algorithms and local buffers are used to prevent several line cards from sending packets onto the backplane at the same time.

Cisco IOS Software: The Brains

Routers aren't just brawn without brains—software is required to provide general control-plane functionality, build tables, and (sometimes) switch packets. Cisco routers run a specialized operating system called Cisco IOS, generally referred to as Cisco IOS Software.

Cisco IOS was originally designed as a small operating system that could be embedded on a single ROM chip, and it has progressed to a full-fledged operating system with memory management, hardware abstraction, process scheduling, and other such services. In the following sections, you look at several aspects of Cisco IOS Software, including how it manages memory, interacts with interface processors, and schedules processes.

Memory Management

In the section "Understanding Router Pieces and Parts," earlier in this chapter, you learned that several types of memory are used—some fast and some slower, depending on what the memory is used for. How does Cisco IOS Software manage this memory? If a process wants to allocate memory in a fast (or shared) area of memory for processing a packet, how does it do so?

When the router boots up, Cisco IOS Software divides memory into pools and regions based on the type of memory used.

Memory Pools

A large variety of memory is available in Cisco routers, depending on the platform; some have just DRAM, others have SDRAM, others have both DRAM and SDRAM, and some have other combinations of memory. [Figure 1-13](#) illustrates some of these memory configurations and shows how they relate to what memory is used for in the router.

Figure 1-13. Memory Types and Their Uses

Storage	Software	Main Memory	Packet Memory	
Flash		RAM		Cisco 2500
Flash		RAM		Cisco 4700
Flash		RAM	SRAM	Cisco 7200
Flash		SRAM	SRAM PCI	Cisco 7200VXR

With all of these different types of memory available, how does IOS divide it up so that processes can simply indicate what they will use the memory for, and get a piece of memory out of the right memory type? Through hardware abstraction.

Hardware abstraction is the process of creating a layer on top of the hardware that looks the same no matter what hardware it's running on. In this case, you want the same types of logical memory—storage, running software, data structures, and packet memory—to be available to processes running under Cisco IOS Software no matter what physical memory is available. The hardware abstraction used here is memory pools. [Figure 1-14](#) illustrates memory pools.

Figure 1-14. Abstracting Memory Types Using Pools

Storage	Software	Main Memory	Packet Memory	Used for
Flash	Processor		I/O	Memory pool
Flash	RAM			Physical memory

When IOS boots on a given router, it determines what sorts of memory are available and builds memory pools as needed. You can examine the layout of the pools on a router by executing the show memory command, as shown in Example 1-1.

Example 1-1. Output of the show memory Command

```
7206-router#show memory
      Head      Total (b)      Used (b)      Free (b)
Processor  61E2EEE0    94179616    12054120    82125496
      I/O    20000000    33554432     224984    33329448
      I/O-2   7800000    8388616     3077528    5311088
--more--
```

The column on the left lists the memory pools available on this router. This particular router is a Cisco 7206, which has two I/O memory pools, because there are two physical sets of fast memory used for storing and processing packets as they are switched through the router.

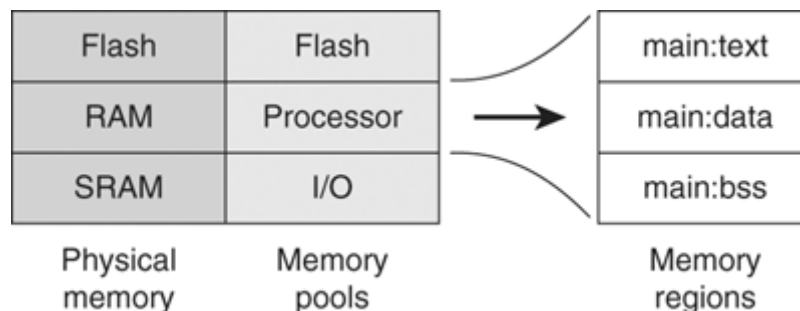
Note

Detailed information about the show commands referenced throughout this chapter are included in the section "Hardware and Software Show Commands," later in this chapter.

Memory Regions

Within memory pools, the memory is subdivided into regions. Each region represents a type of memory within the pool. For example, within the processor pool, some memory is used for general data storage (the heap) and other memory is used for executing code (text). Figure 1-15 illustrates these regions within the context of memory pools.

Figure 1-15. Memory Regions



You can see what region each memory pool has been divided into by using the show region command, as shown in Example 1-2.

Example 1-2. Output of the show region Command

```
7206-router#show region
Region Manager:
      Start      End      Size (b)  Class  Media  Name
0x07800000  0x07FFFFFF  8388608  Iomem  R/W    iomem2
0x20000000  0x21FFFFFF  33554432 Iomem  R/W    iomem
0x57800000  0x57FFFFFF  8388608  Iomem  R/W    iomem2:(iomem2_cwt)
0x60000000  0x677FFFFFF 125829120 Local  R/W    main
0x60008960  0x611682CB  18217324 IText  R/O    main:text
0x6116A000  0x61B4E17F  10371456 IData  R/W    main:data
0x61B4E180  0x61E2EEDF  3018080  IBss   R/W    main:bss
0x61E2EEEE0 0x677FFFFFF  94179616 Local  R/W    main:heap
0x70000000  0x71FFFFFF  33554432 Iomem  R/W    iomem:(iomem_cwt)
0x80000000  0x877FFFFFF 125829120 Local  R/W    main:(main_k0)
0xA0000000  0xA77FFFFFF 125829120 Local  R/W    main:(main_k1)
```

Each region can also have subregions, which are indicated by a colon followed by the subregion name. For example, main has four subregions: main:text, main:data, main:bss, and main:heap. Each of these subregions is used for different types of data. Main:text is marked as read only (R/O in the Media column), because that is where the executing code is stored, and the executing code should not be overwritten. Main:data, however, has been marked as read/write (R/W in the Media column), because this is where general initialized data is kept.

Each region can also have several aliases, which are used to access the memory in different ways. Aliases are indicated by a region name followed by a colon, and then the alias name in parentheses, such as main:(main k0). This can represent a cached versus an uncached view of the same memory, different access methods, and so on.

Packet Buffers

Packet buffers are where packets are stored while they are being switched (on some platforms) or processed in some way. Packet buffers are created out of the iomem memory region and are managed by a buffer management process.

Types of Buffer Pools

Packet buffers are kept in pools of buffers; there are three ways in which a packet buffer pool can be classified:

- Private or public— Public packet buffer pools contain buffers that can be taken by any process on the router, while only certain processes (generally an interface device driver) can take buffers from a private buffer pool.
- Dynamic or static— Dynamic buffer pools can change size as they are used; frequently used dynamic pools can increase in size, whereas infrequently used pools can shrink. Static buffer pools are created at a fixed size and remain that size throughout their life.
- Size— Buffers available from the pool occur in six sizes: small (up to 104 bytes), middle (up to 600 bytes), big (up to 1524 bytes), very big (up to 4520 bytes), large (up to 5024 bytes), and huge (up to 18,024 bytes).

Packet Buffer Sizes: You might be wondering why packet buffers are such odd sizes. Each packet buffer size is designed to handle packets of a common size. For example, small buffers have enough space in them for 64-byte packets plus the overhead of any Layer 2 (MAC) headers. Others are set up for the maximum transmission unit size of a given link type. Big buffers fit Ethernet packets plus any Layer 2 data. The huge buffers are so much larger than the largest maximum transmission unit (16,384 bytes) for terminating tunnels and other sorts of virtual interfaces, where the packet must be reassembled in the packet buffer before it can be switched.

Private buffer pools are normally attached to an interface; in the Cisco IOS Software command show buffers, they are normally called interface buffers. [Example 1-3](#) shows output from the show buffers command.

Example 1-3. Interface Buffer Pools in the Output of the show buffers Command

```
2651-router#show buffers
....
Interface buffer pools:
CD2430 I/O buffers, 1524 bytes (total 0, permanent 0):
    0 in free list (0 min, 0 max allowed)
    0 hits, 0 fallbacks
....
```

[Example 1-4](#) shows public buffers in the show buffers command.

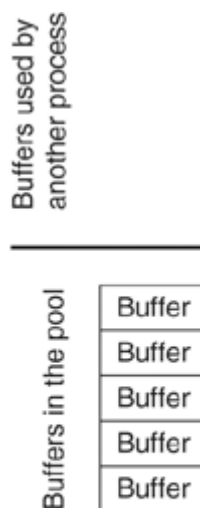
Example 1-4. Public Buffer Pools in the Output of the show buffers Command

```
2651-router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 5, permanent 5):
    2 in free list (2 min, 5 max allowed)
    3 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
```

Managing Buffer Pools

To explain buffer pool management, this section examines a series of operations on a buffer pool as an example. You'll examine the output of the show buffers command for the small public buffer pool through a series of buffers being consumed for packets received, packet buffer creation, and so on. [Figure 1-16](#) shows the beginning state of a packet buffer.

Figure 1-16. Buffer Pool Management, Step 1



Step 1

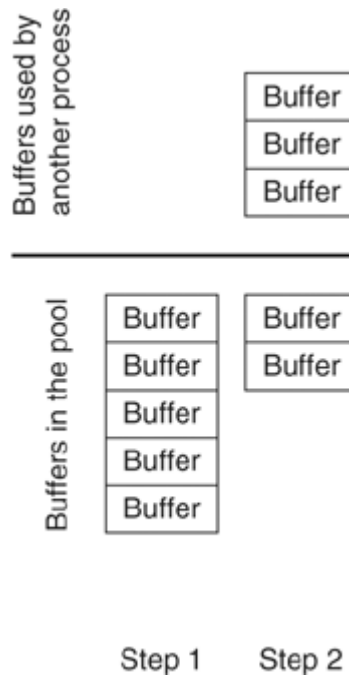
Example 1-5 shows that the buffer pool starts with five buffers in the pool, which is the number of permanent buffers. All the other counters start at 0.

Example 1-5. Public Buffer Pool from the show buffers Command in Initial State

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 5, permanent 5):
    5 in free list (2 min, 5 max allowed)
    0 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
```

A process pulls three buffers from the pool, as illustrated in Figure 1-17.

Figure 1-17. Buffer Pool Management, Step 2



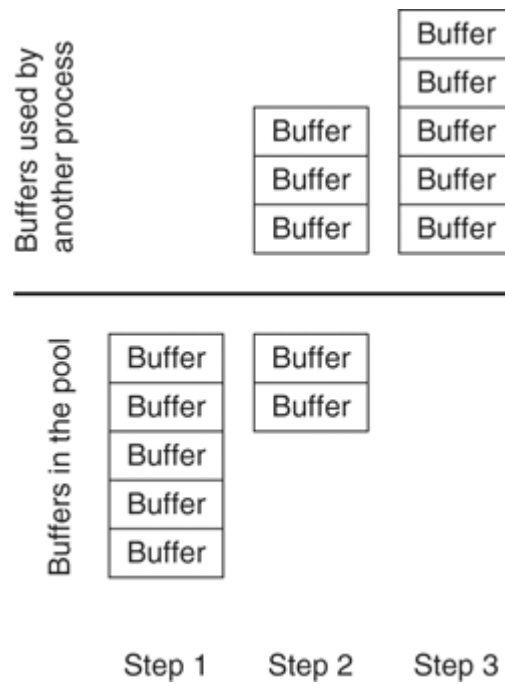
Example 1-6 shows that the number of buffers now in the free list is two, and there are three hits, because three buffers were used from this pool.

Example 1-6. Public Buffer Pool After Three Hits

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 5, permanent 5):
    2 in free list (2 min, 5 max allowed)
    3 hits, 0 misses, 0 trims, 0 created
    0 failures (0 no memory)
```

The same process now uses two more buffers from the same pool; the result is illustrated in Figure 1-18.

Figure 1-18. Buffer Pool Management, Step 3



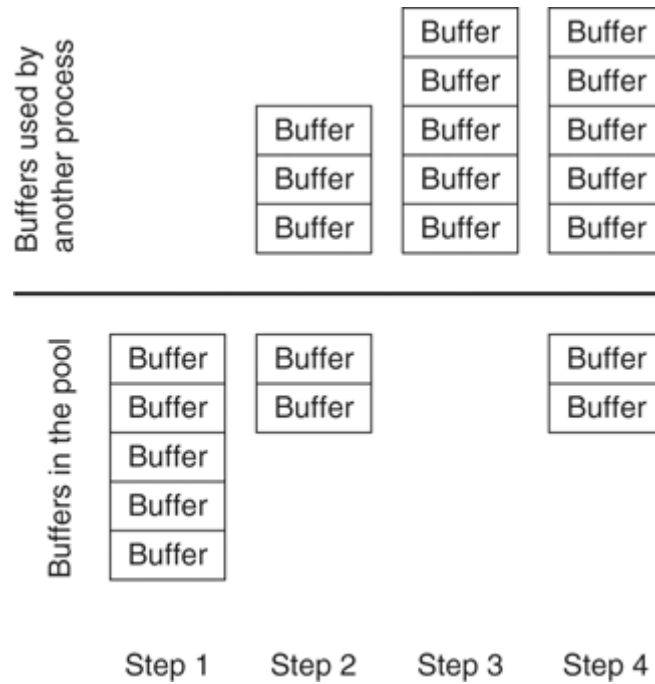
Example 1-7 now shows five hits, because five buffers have been pulled from this pool successfully, and none are in the free list. Each buffer pulled from the pool in this step has dropped the number of free buffers in the pool below the minimum free allowed, so each one is counted as a miss (even though the buffer was supplied to the other process successfully).

Example 1-7. Public Buffer Pool After Five Hits

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 5, permanent 5):
  0 in free list (2 min, 5 max allowed)
  5 hits, 2 misses, 0 trims, 0 created
  0 failures (0 no memory)
```

The pool manager now runs and notes that this buffer pool has fewer free buffers than it is allowed. So it creates two more buffers so that the pool has at least the minimum number of free buffers allowed. This is illustrated in Figure 1-19.

Figure 1-19. Buffer Pool Management, Step 4



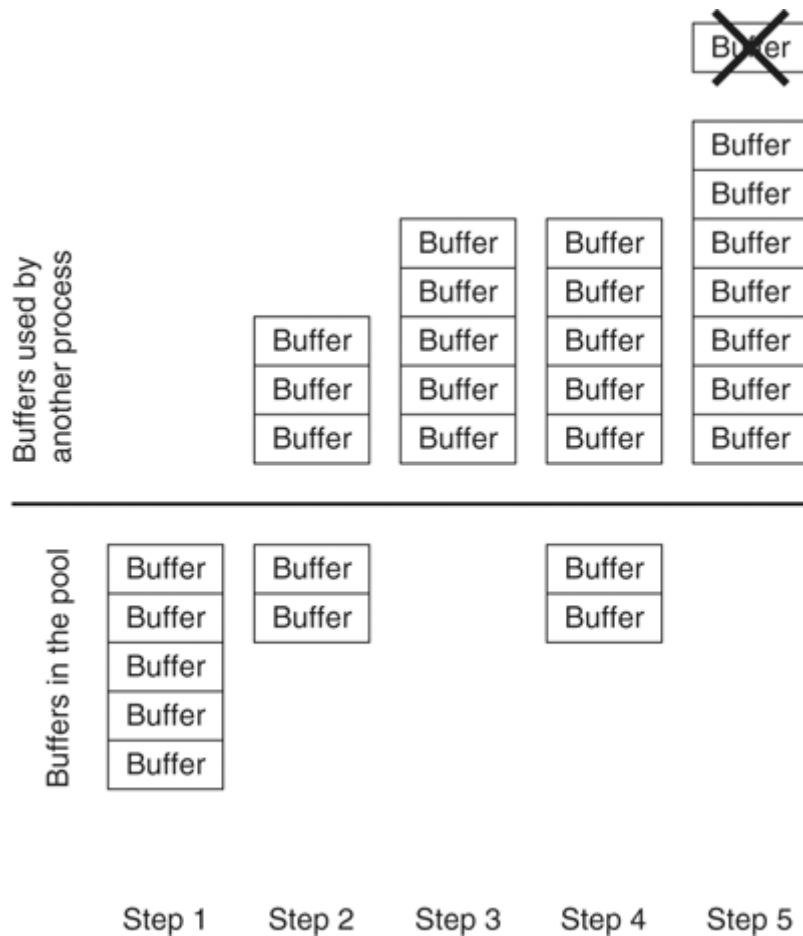
Example 1-8 now shows two buffers in the free list and two created buffers, because the pool manager had to create two buffers to reach the minimum free allowed.

Example 1-8. Public Buffer Pool After Two Are Created

```
Public buffer pools:
Small buffers, 104 bytes (total 7, permanent 5):
  2 in free list (2 min, 5 max allowed)
  5 hits, 2 misses, 0 trims, 2 created
  0 failures (0 no memory)
```

The other process now requests three more buffers; the results are shown in Figure 1-20.

Figure 1-20. Buffer Pool Management, Step 5



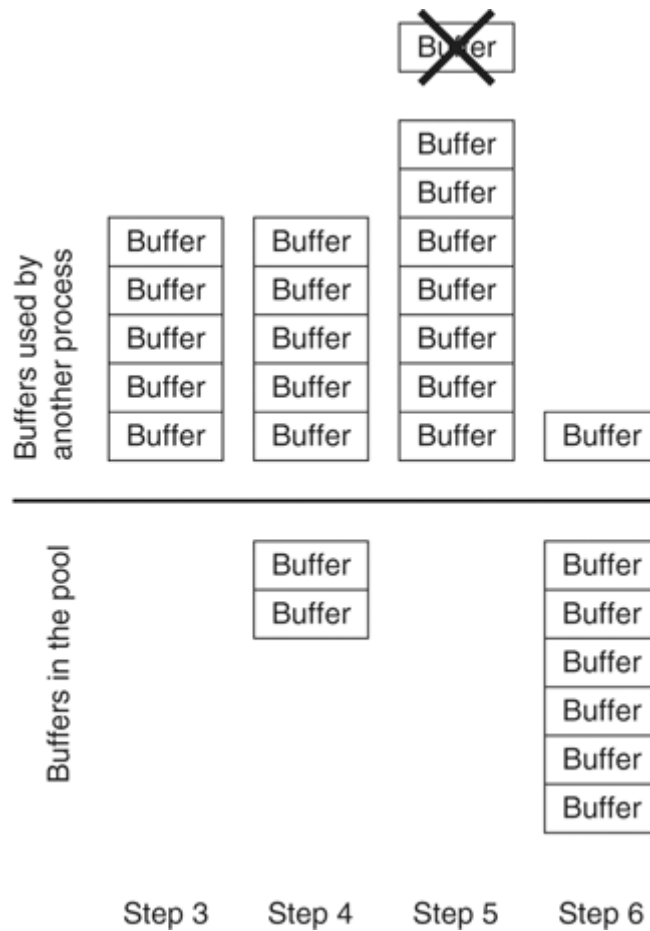
Example 1-9 shows that the pool is now back down to 0 in the free list. Because two more buffers were successfully pulled from the pool, the output shows seven hits. Because the two buffers used from the pool each dropped the number of buffers in the free list below the allowed minimum number of free buffers, the output shows five misses. The third miss added is because of the buffer that the pool failed to supply; this is also counted as a miss.

Example 1-9. Public Buffer Pool After Five Misses

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 7, permanent 5):
  0 in free list (2 min, 5 max allowed)
  7 hits, 5 misses, 0 trims, 2 created
  1 failures (0 no memory)
```

The process that is using these seven small buffers returns all but one of them to the pool. The result is illustrated in Figure 1-21.

Figure 1-21. Buffer Pool Management, Step 6

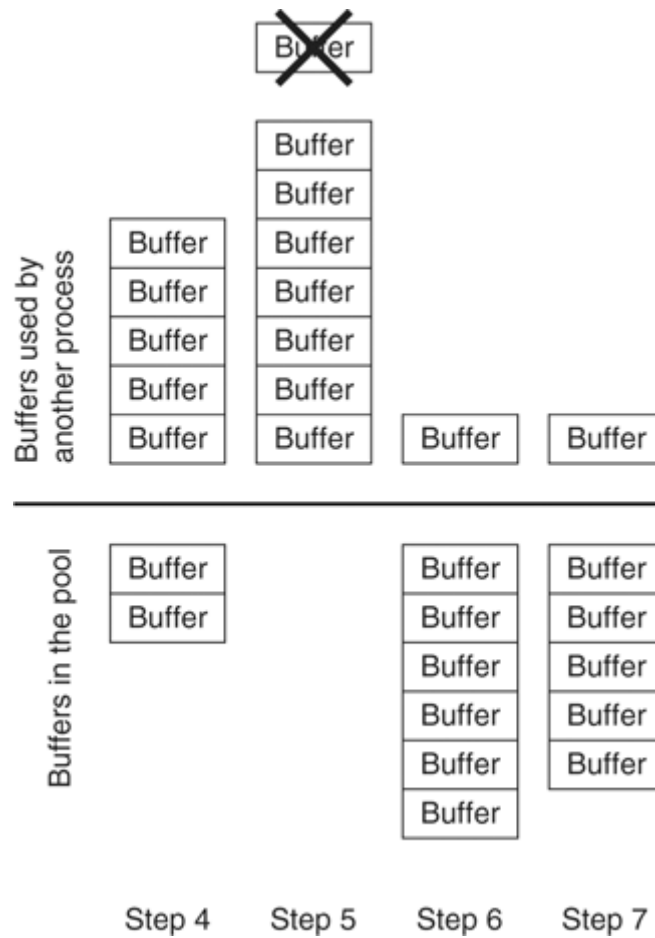


Example 1-10 shows six buffers in the free list, but only five are allowed to be in the free list. The next time the pool manager runs, it will trim one of these buffers, which essentially means that it will release the memory associated with it. The result is shown in Figure 1-22.

Example 1-10. Public Buffer Pool After a Trim

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 7, permanent 5):
  6 in free list (2 min, 5 max allowed)
  7 hits, 5 misses, 0 trims, 2 created
  1 failures (0 no memory)
```

Figure 1-22. Buffer Pool Management, Step 7



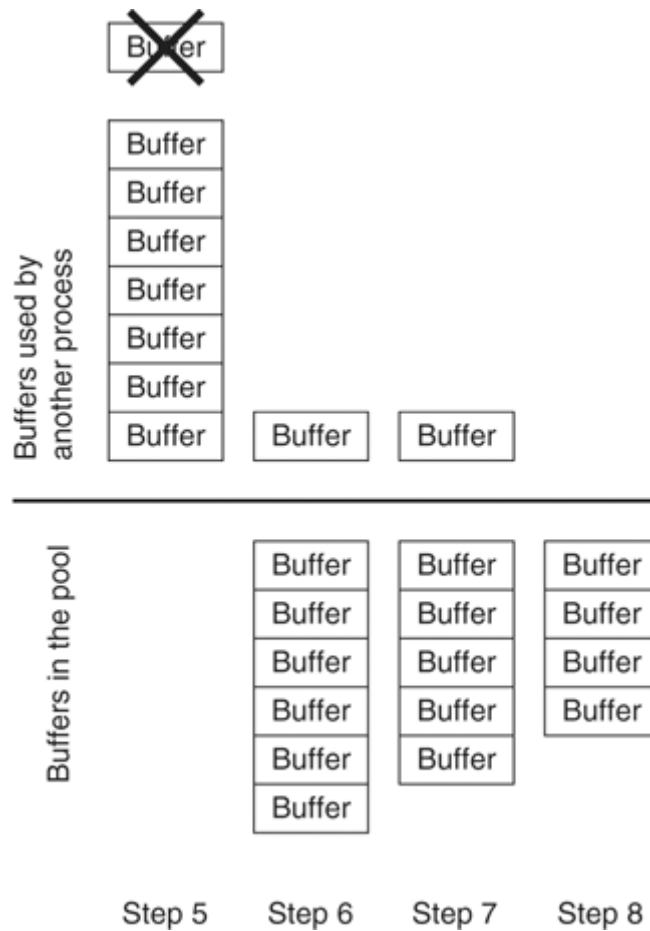
Example 1-11 shows that because the pool manager has now trimmed one buffer from this pool, the free list is down to five and one trim is recorded.

Example 1-11. Public Buffer Pool After Final Trim

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 7, permanent 5):
  5 in free list (2 min, 5 max allowed)
  7 hits, 5 misses, 1 trims, 2 created
  1 failures (0 no memory)
```

Finally, the other process returns the last buffer, which will again bring the free list up to six buffers. The pool manager will again trim this to five buffers, because the maximum free is five buffers, and the number of permanent buffers in the pool should be five. The result is illustrated in Figure 1-23.

Figure 1-23. Buffer Pool Management, Step 8



The output of the show buffers command in [Example 1-12](#) shows this public buffer pool in its final state.

Example 1-12. Public Buffer Pool in Final State

```
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 7, permanent 5):
    4 in free list (2 min, 5 max allowed)
    7 hits, 5 misses, 2 trims, 2 created
    1 failures (0 no memory)
```

Interaction with Interface Processors

In this chapter, you've learned how interface processors copy incoming packets from the network media to locations in the router's memory (either shared memory or memory that is local to a line card). You learn how the packet is switched after it is copied into the router's memory in [Chapter 2](#), "Cisco Express Forwarding."

But after the packet is in memory, device drivers notify the switching engine that there is a packet to be switched. Device drivers are the critical pieces of code that connect the interface processors to the switching path in the router. Device drivers generally do several things:

- Classify packets as they are received so that the right switching code is called to process the packet (IP, IPX, AppleTalk, and so on)
- Make certain that the receive and transmit rings used by the interface processors to copy packets don't run out of space (if possible)
- Program the interface controller with any needed information, such as which Layer 2 (MAC) multicast addresses the interface processor should be listening to

Processes and Scheduling

Cisco IOS Software, like all operating systems, uses processes—individual "applications" that perform certain operations within the context of the operating system. Some of the processes that run within Cisco IOS Software include:

- The pool manager, which manages the memory pools described in the section "[Memory Management](#)," earlier in this chapter.
- IP input, which processes Internet Protocol (IP) packets that are not switched in some other way. More information on this is contained in the section "[Putting the Pieces Together: Switching a Packet](#)," later in this chapter.
- Each of the routing protocols, such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Enhanced Interior Gateway Routing Protocol (EIGRP), and Intermediate System to Intermediate System (IS-IS), has one or more processes each.

You'll learn about some of the specific processes that run within Cisco IOS Software in later sections.

Process Memory

In some operating systems, each process is given a block of memory that other processes in the system cannot access. Each process is protected from every other process, because no process can write in another process's memory space. Cisco IOS Software, however, uses one flat memory space for all processes, although it does protect some memory space from being written into (by marking it read only through the pool manager, as noted in the section "[Memory Management](#)," earlier in this chapter).

Both models have advantages and disadvantages. For example, in operating systems where processes are protected from writing into each other's space, one process cannot corrupt another process's memory, so the system might not experience as many failures because of memory corruption. On the other hand, operating systems that use a flat memory model allow processes to share information easily and quickly.

Because almost all Cisco IOS Software was originally written with simplicity and speed as the top priorities, a flat memory model was chosen. The `show processes memory` command, which is explained in the section "[Hardware and Software Show Commands](#)," later in this chapter, provides information about the memory being used by each process.

Process Scheduling

If there are multiple processes running, and only one can run on the processor at any time, the scheduler determines which process runs at any given time. The following sections look at how the scheduler works by examining the scheduler itself, the life cycle of a process, process priorities, and how the scheduler decides which process should be run at any given time. These sections also examine the process watchdog.

Understanding the Scheduler

To understand the scheduler, you need to begin by noting that Cisco IOS Software is a nonpreemptive multitasking system, which means that when a process begins running, the process itself decides when to release the processor (within limits, as discussed in the section "[Process Watchdog](#)," later in this chapter).

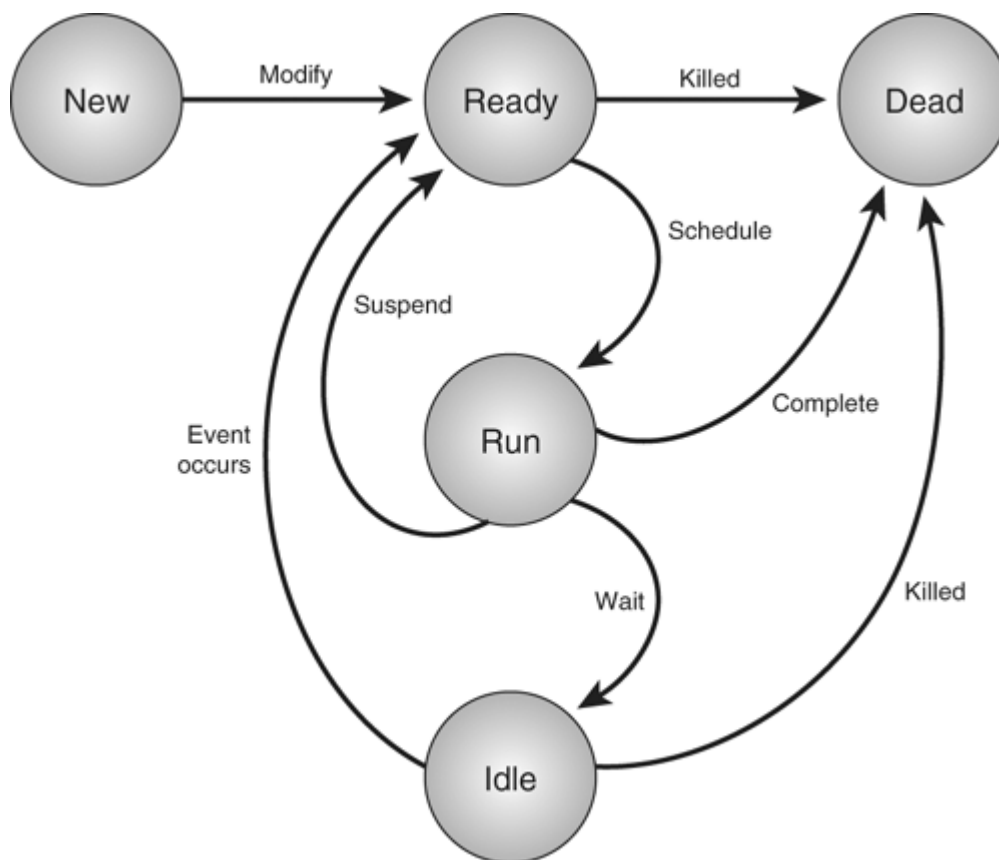
The scheduler itself, then, should be lightweight, primarily responsible for ordering process priorities, determining which processes should be moved into which states periodically, and determining the next process to be run after a process releases the processor.

Although Cisco IOS includes a scheduler, it does not include a scheduler process; in other words, the scheduler doesn't exist as a stand-alone process that gains access to the CPU from time to time. Instead, the scheduler simply runs each time a process releases the processor. It first checks to see whether any processes should change state and then determines which process should run next.

Process Life Cycle

Figure 1-24 illustrates the life cycle of a process within Cisco IOS Software.

Figure 1-24. Life Cycle of a Process in Cisco IOS Software



A process within Cisco IOS Software can be in one of five states:

- New— The process has just been created and has no resources. It has never been scheduled to run.
- Ready— The process is ready to run and should be scheduled on the processor the next time it is eligible.
- Run— The process is currently running.
- Idle— The process is waiting for some event to become ready to run.
- Dead— The process is dead; it is being cleaned up.

The details of the life cycle processes within Cisco IOS Software are as follows:

1. Processes are created in the new state; at this point, the process doesn't have any resources, nor is there any way for the process to be scheduled to run. When you first enter a routing protocol subconfiguration mode, for example, the protocol's routing process is created in the new state.
2. After a process is created, it is modified and placed in the ready state. Modification primarily entails assigning resources to the process, such as initialized and uninitialized data segments, access to the console, and so on. Processes in the ready state are eligible to be scheduled for execution on the processor.
3. From the ready state, the process is scheduled and moved to the run state. There is no way for a process that is in the ready state to move directly to the idle state; it must run to become idle.
4. A process that is running can complete, which means that it reaches the end of its task and finishes. The next time this process needs to be run, it will need to be re-created, modified, placed in the ready state, scheduled, and finally run. This occurs when a routing protocol is removed from the router, for example. Some show commands issued from the enable prompt create processes that only run until the show output is generated; they then complete and move into the dead state.
5. A process that is running can suspend, which means that it has more work to do, but is at a point at which it can quit processing for some time and allow other processes that might be waiting to be scheduled and placed in the run state. Processes that suspend are placed in the ready state, which means that they are eligible to be scheduled and run the next time the scheduler runs.
6. A process that is running can also wait, which means it finds that it cannot process any more information until some piece of information is available, or some other process runs and does some work on which it is dependent. One example of this is when a routing protocol process finishes processing all the packets that the lower-layer protocols have passed to it, and the routing protocol process has no further work to do. The routing protocol process would then wait and enter the idle state.
7. A process in the idle state can be killed, for example, when the user removes the protocol or prematurely ends the process by interrupting some output. Processes in the idle state that are killed are moved into the dead state.
8. If a process in the idle state is waiting on data that becomes available, or if an event occurs that indicates the process has new work to do, the process will be moved to the ready state, where it is eligible to be scheduled and then run.

The current state of a process is indicated in the output of the show processes enable command, as shown in [Example 1-13](#).

Example 1-13. Sample Output from the show processes Command

Code View: [Scroll](#) / [Show All](#)

```
7206#show processes
CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%
  PID QTy      PC Runtime (ms)   Invoked   uSecs      Stacks TTY Process
   2 M*          0           8         86         93 9888/12000  0 Exec
   3 Lst 60655C58      345736    129733    2664  5740/6000  0 Check
heaps
....
```

The Ty columns in this output, highlighted in [Example 1-13](#), indicate the process state, as follows:

- *— Process is running on the CPU (you'll almost always see exec as the current process on the processor, because it is almost always the place you run this command from).

As [Figure 1-25](#) illustrates, there are actually four queues of ready-to-run processes (or four ready queues), rather than one—one for each process priority.

The way in which processes are scheduled and run, as well as moved between the various queues, is explained in the following list.

1. Process A begins in the idle state, waiting on some system event before it needs to run again. When the scheduler runs, it finds that the event that Process A was waiting on has occurred, so it moves the process from the idle queue to the high-priority ready queue.
2. Now that Process A has been moved from the idle queue to the high-priority ready queue, the scheduler needs to decide what process should be run. It begins with the critical ready queue and finds that Process B is ready to run, so it schedules Process B.
3. While Process B is running, it finds that it needs to wait until some system event occurs to complete further processing, and Process B is placed on the idle queue.
4. The scheduler now examines the critical queue again and finds that Process D is ready to run. Process D is scheduled.
5. Process D uses the processor for some time and finds that it needs to wait for a system event as well before continuing to process, so the scheduler places Process D in the idle queue.
6. The scheduler now examines the critical queue and finds it is empty, so it examines the high-priority ready queue to see whether any high-priority processes are ready to run. It finds Process A ready to run in the high-priority ready queue, so it schedules it.
7. Process A runs for some time and finds it needs to wait on some other event or piece of information, so the scheduler moves it back to the idle queue.
8. The scheduler now examines the high-priority ready queue and finds no processes ready to run, but it does find Process C on the medium-priority ready queue; it schedules this process to run next.

Using this example, the following rules summarize the scheduler's operation:

- The scheduler runs as each process releases the processor.
- Each process in the idle and waiting state is examined; if an event has occurred that a process was waiting on, the process will be moved to the appropriate ready queue (based on its priority).
- If there are any processes in the critical-priority ready queue, one of them is scheduled and the scheduler releases the processor.
- If no critical processes are ready to run, the high-priority ready queue is examined. If high-priority processes are ready to run, one is scheduled and the scheduler releases the processor.
- If no critical- or high-priority processes are ready to run, the medium-priority ready queue is checked. If medium-priority processes are ready to run, one of them is scheduled to run, and the scheduler releases the processor.
- If no critical-, high-, or medium-priority processes are ready to run, the low-priority ready queue is checked.

Process Watchdog

If a process runs until it is ready to release the processor, can't it run forever, keeping any other processes from running? No, because each process is carefully watched by the watchdog timer to make certain that it doesn't run

for too long. Every 4 milliseconds, the watchdog timer interrupts the processor, adjusts the system clock, and does other background tasks.

While doing these things, the watchdog also checks to see what process is running on the processor. If it finds a single process running on the processor for more than 2 seconds, it flags the process as a processor hog (also called a CPU hog). The next time the scheduler runs, it notes that the process releasing the processor has been flagged as a CPU hog, and prints a diagnostic message that the Cisco Technical Assistance Center (TAC) can use to determine which process ran too long.

If a process runs for 4 seconds, it will be terminated.

Special Processes

There are three special processes shown in the show processes memory command, but not in the show processes command. [Example 1-14](#) shows these processes.

Example 1-14. Special Processes in the Output of the show processes memory Command

```
router#show processes memory
Total: 35629228, Used: 5710380, Free: 29918848
  PID TTY   Allocated      Freed   Holding   Getbufs   Retbufs Process
   0   0       74032         1808   3358588     0         0 *Init*
   0   0         1116        275416     1116     0         0 *Sched*
   0   0 1290760868 1283157048 1223504   164928     0 *Dead*
```

What do the Init, Sched, and Dead processes do, and why don't they show up in show processes? They don't show up in show processes because they aren't processes in the proper sense. They are simply markers to indicate where certain pieces of memory are being used, as follows:

- **Init**— Accounts for the memory allocated before the scheduler, memory pool manager, and other such processes come up.
- **Sched**— Accounts for the memory used in the process scheduler; the scheduler isn't a separate process.
- **Dead**— Accounts for memory that is allocated to processes that have entered the dead state.

Putting the Pieces Together: Switching a Packet

Now that you've examined all the individual pieces used in a Cisco router to switch packets, the following sections put them all together so that you can see how they interact.

Getting the Packet off the Network Media

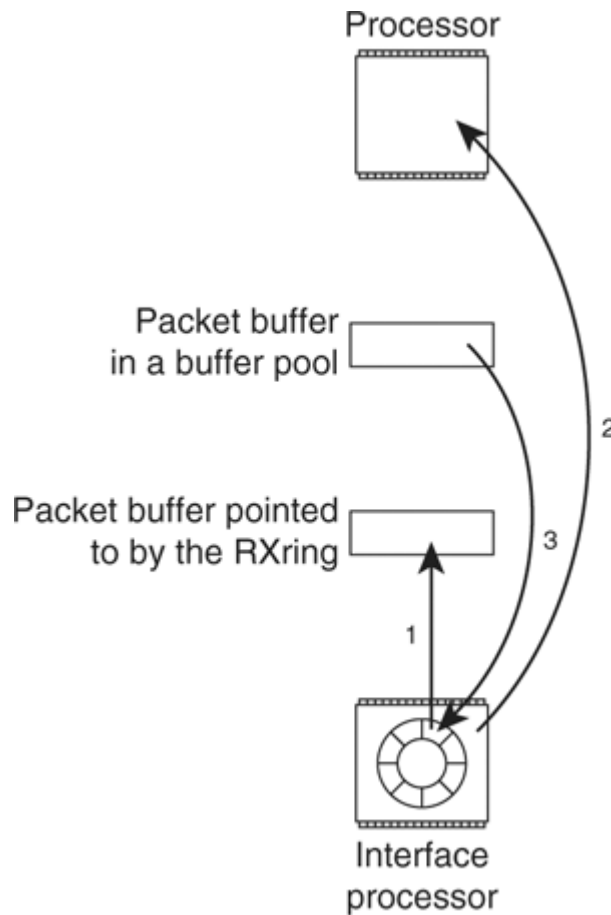
Beginning with the packet being handled by the interface processor, there are a bewildering number of ways in which a packet is copied off the network media. The method used depends on the hardware platform involved. Three different general types of platforms are discussed in the following sections:

- **Shared memory**
- **Line cards with a centralized processor**
- **Line cards with distributed switching**

Inbound Packets on Shared Media Platforms

The inbound packet path on a shared memory platform, such as the Cisco 2600, is illustrated in [Figure 1-26](#).

Figure 1-26. Inbound Packet Handling on Shared Media Platforms



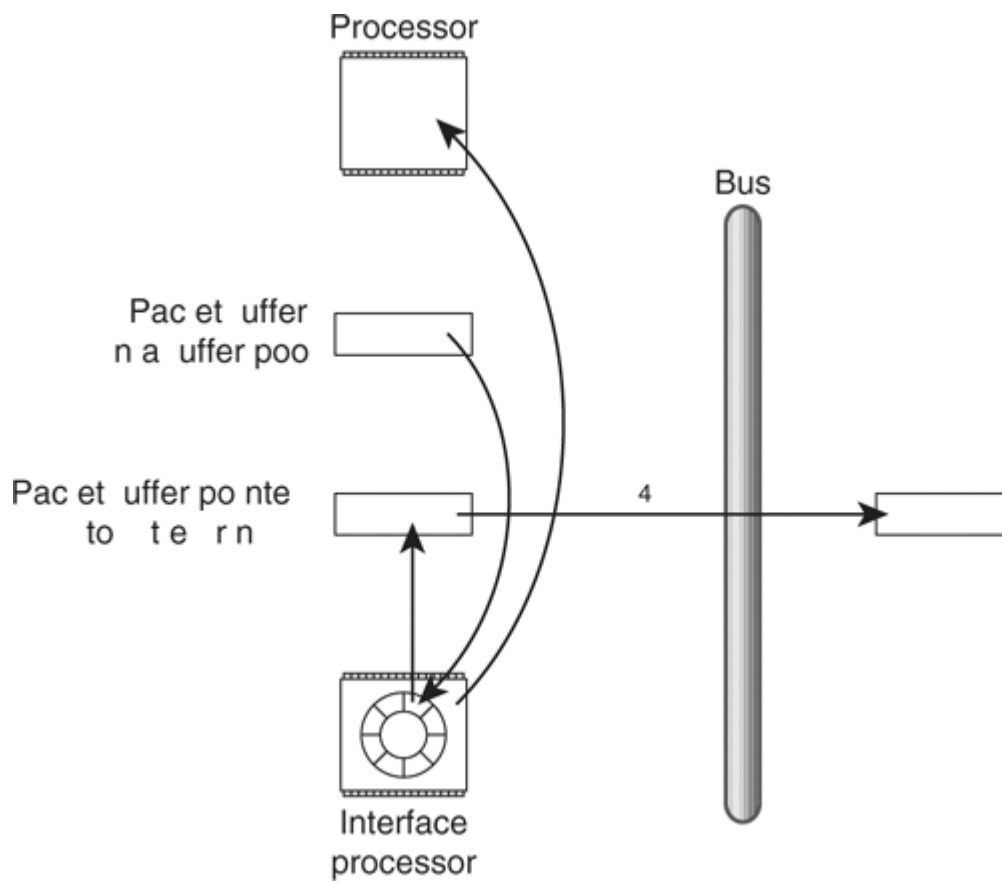
The following list explains the process of handling inbound packets on shared media platforms:

1. The interface process detects the optical or electrical signals that indicate a new packet is arriving on the interface. The Layer 2 address is read and checked to make certain that the router should receive and process this packet, and the packet is copied into the packet buffer pointed to by the next receive (RX) ring entry in the interface processor.
2. After the packet has been copied into the memory location pointed at by the RX ring, the interface processor interrupts the processor in the router to indicate a packet has been received. This is called the receive interrupt.
3. The device driver now runs and immediately reprograms the RX ring so that it points at a packet buffer from some other memory pool. The packet buffer, now used by the RX ring, is removed from the pool of packet buffers. The device driver also classifies the packet and determines which process should switch it.

Inbound Packets on Centralized Switching Routers with Line Cards

Routers with line cards and that perform switching on a centralized processor, such as the Cisco 7200 series of routers, need to pass packets received on their interfaces through one more step before being able to switch them. Packets must be moved across the bus that connects the line cards to the centralized processor, as illustrated in [Figure 1-27](#).

Figure 1-27. Inbound Packets on Centralized Switching Routers with Line Cards



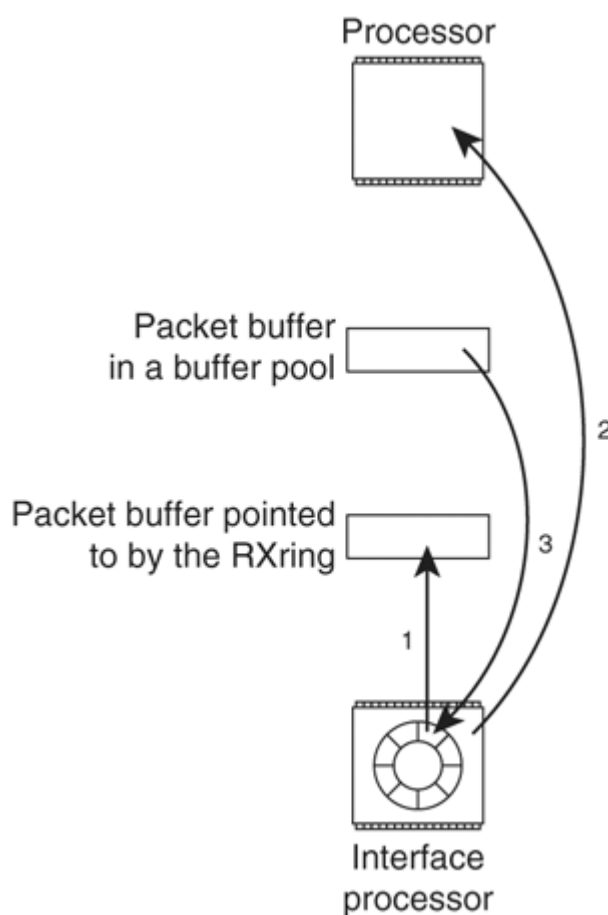
The following list explains the process:

1. The interface process detects the optical or electrical signals that indicate a new packet is arriving on the interface. The Layer 2 address is read and checked to make certain that the router should receive and process this packet, and the packet is copied into the packet buffer pointed to by the next receive (RX) ring entry in the interface processor.
2. After the packet has been copied into the memory location pointed at by the RX ring, the interface processor interrupts the processor in the router to indicate a packet has been received. This is called the receive interrupt.
3. The device driver now runs and immediately reprograms the RX ring so that it points at a packet buffer from some other memory pool. The packet buffer, now used by the RX ring, is removed from the pool of packet buffers.
4. The device driver copies the packet across the bus so that it is in memory on the route processor. The device driver also classifies the packet and determines how it should be switched.

Inbound Packet Handling on Distributed Switching Platforms

The inbound packet handling path on platforms that switch packets on the line cards themselves, such as the Cisco 12000 and 7500 series of routers, is illustrated in [Figure 1-28](#).

Figure 1-28. Inbound Packet Handling on Distributed Switching Platforms



The following list explains the process of handling packets on distributed switching platforms:

1. The interface processor detects the electrical or optical signals that indicate a packet is being received. The interface processor checks the Layer 2 address (MAC address) of this packet to see whether the router needs to receive or process it. As the packet is received, it is copied into a packet buffer on the line card based on the pointer contained in the receive ring.
2. The interface processor now interrupts the processor on the line card to notify it that a packet has been received. This is the receive interrupt.
3. The device driver runs during this interrupt; it immediately replaces the packet buffer on the receive ring (by reprogramming the interface processor) and then classifies the packet to determine how it should be switched.

As you can see from the illustration, the process on all three types of platforms is similar; the primary difference is which processor is interrupted to notify the router of a new packet that needs to be processed, and where the packet sits in memory when the actual switching of the packet begins.

Switching the Packet

Now that the packet has been copied off the network media into a packet buffer on the router, it can be switched. A Cisco router can use two basic methods to switch packets:

- Hardware-based switching
- Software-based switching

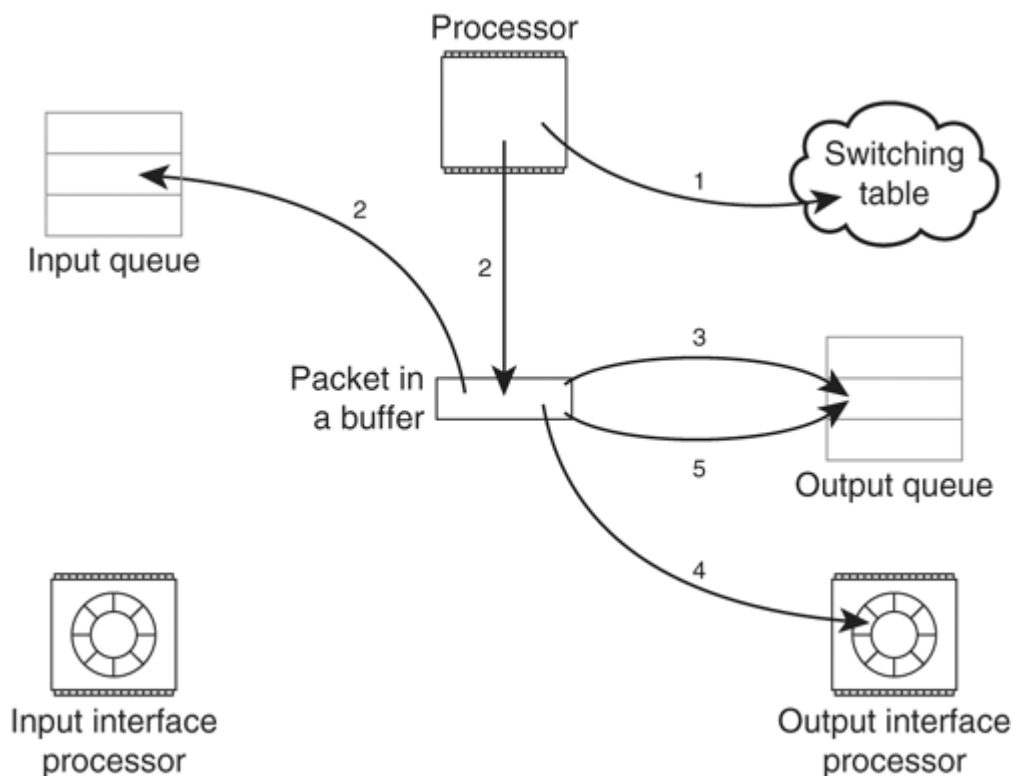
Hardware-based switching exists in many different forms on Cisco routers, from the Cisco 12000 series and Cisco 6500 series, which use custom-designed ASICs, to the Cisco 10000, which uses a programmable ASIC called the Toaster to perform Parallel Express Forwarding (PXF). The following sections focus on software packet switching.

Switching the Packet Quickly During the Receive Interrupt

Recall from the preceding section that when a packet is received and copied into memory, the processor is interrupted and the device driver code runs. After the device driver has run, the switching code runs while the processor is still in the context of this receive interrupt.

Figure 1-29 illustrates the steps taken to software-switch the packet while it is still within the receive interrupt.

Figure 1-29. Software-Switching a Packet in the Receive Interrupt



The following list explains the process:

1. The switching code consults the switching table to determine whether the packet's destination is known, the next hop's MAC header is known, and the outbound interface is known.
2. If the information needed to switch the packet is available in the switching table, and the packet is not destined for the router itself, the packet header is modified. If the information needed to switch the packet is not available in the switching table, or if the packet is destined to the router itself, the buffer the packet is in is placed on the input queue of the appropriate process to be dealt with later. Note that the packet itself is not moved; instead, a pointer to the packet buffer that the packet was originally copied into is placed on the input queue of the appropriate process. The next section in this chapter explains what happens to the packet after it is placed on the input queue of a process.
3. After the MAC header is rewritten, the output queue of the outbound interface is checked. If any packets are in the output queue of the outbound interface, a pointer to the buffer that the packet is in is copied into this output queue.

4. If no other packets are on the output queue of the outbound interface, the transmit ring of the outbound interface is then checked. If there is space on the transmit ring of the outbound interface, a pointer to the buffer containing the packet is placed on the transmit ring.
5. If the transmit ring is full, a pointer to the buffer containing the packet is placed on the output queue of the outbound interface.

After the packet has been moved into the appropriate place for further processing—the input queue of some process, the output queue of the outbound interface, or the transmit ring of the outbound interface—the receive interrupt is released, and the processor continues running the process interrupted by the received packet.

Out-of-Order Packets and Fancy Queuing

These last three steps might seem odd when you first read them—Why would the router check the output queue for packets before deciding whether to transfer the packet to the transmit ring? To prevent out-of-order packets and to make fancy queuing mechanisms, such as weighted fair queuing (WFQ), do their job.

Let's start with out-of-order packet prevention. Assume that a single packet is received and is placed in the input queue of some process for later processing. As this packet is processed, it builds the information needed to switch future packets of this type and adds it to the switching table. The packet is finally placed on the output queue for the outbound interface.

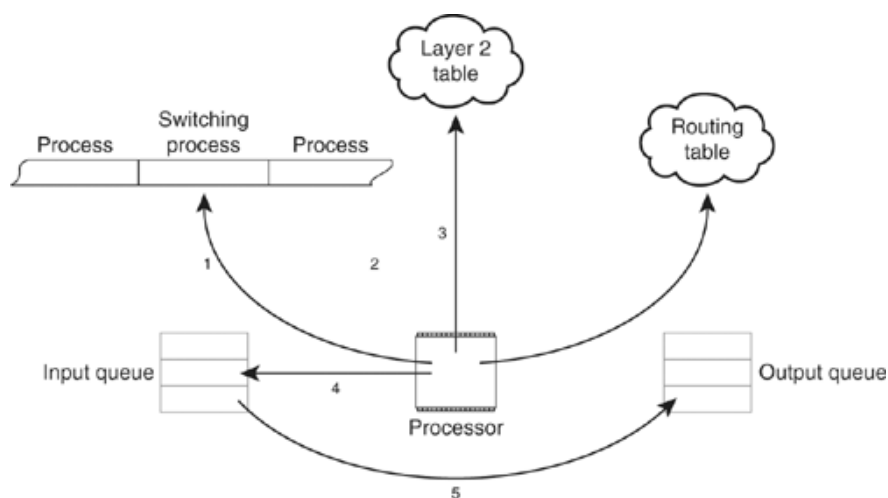
Another packet now arrives of this same type, and during the receive interrupt, switching information for this packet is found in the switching table. If the router now uses this information to switch the packet, and places it directly on the transmit ring of the outbound interface, the second packet will be transmitted before the first. To prevent this sort of error, the output queue of the outbound interface is checked before placing packets directly on the transmit ring of the outbound interface.

Fancy queuing of any type would also be ineffective if all the packets being switched during the receive interrupt were placed directly on the transmit ring of the outbound interface—there wouldn't be any way to make certain that packets which should be transmitted first really will be, because all fancy queuing takes place between the output queue and the transmit ring of the interface.

Process-Switching the Packet

Figure 1-30 illustrates what happens when a packet is placed on the input queue of some process to be switched or otherwise processed.

Figure 1-30. Process-Switching a Packet



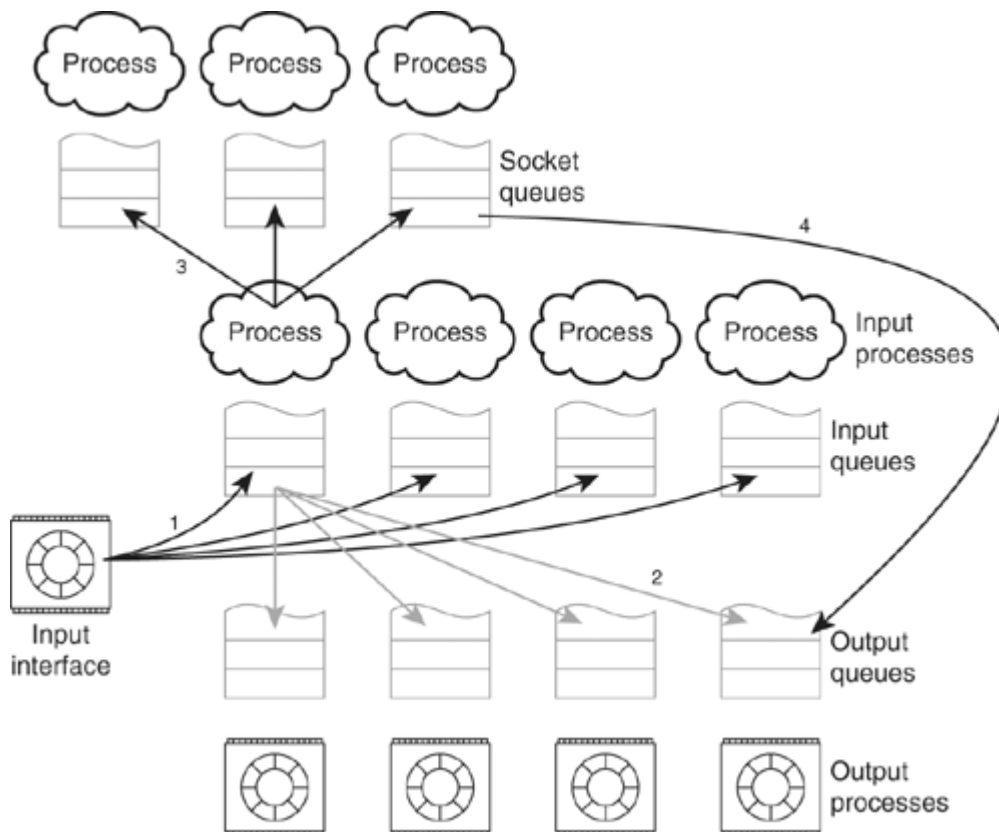
The following list explains process-switching a packet:

1. The processor finishes the currently running process and turns control over to the scheduler. The scheduler notes that there is a packet on the input queue of one of the switching processes and moves the process into the ready queue. The process is scheduled and runs.
2. The process examines the packet header and looks in the routing table to determine the outbound interface and next hop.
3. The switching process looks up the next hop in the Layer 2 tables to determine what Layer 2 (MAC) header to use when forwarding the packet.
4. The switching process rewrites the Layer 2 (MAC) header.
5. The packet is removed from the input queue of the switching process and placed on the output queue of the interface. (The packet isn't actually moved; the pointer to the buffer the packet is in is moved.)

Input and Output Queues

Two queues are mentioned here that aren't explained elsewhere: the input queue and the output queue. There are multiple input and output queues within Cisco IOS Software. [Figure 1-31](#) illustrates these queues.

Figure 1-31. Input and Output Queues



Each process that either switches or otherwise handles packets has an independent input queue, and each interface that can transmit packets has an independent output queue. As packets are received that either must be switched by one of the switching processes or must be processed by the router, they are placed in the input queue for the appropriate process, as shown with the line marked 1 in

Figure 1-31. After they have been switched by the correct process, they are placed in the output queue of the output interface, as shown by line 2.

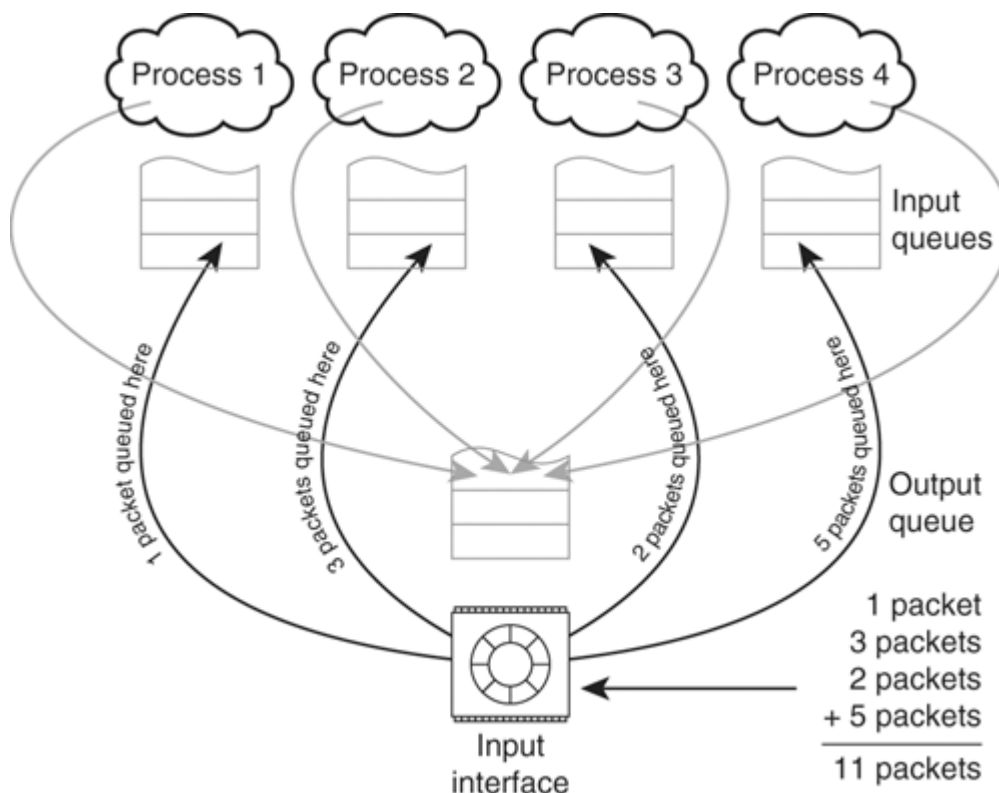
Packets that are destined to the router, such as routing protocol control packets and echo requests, are passed into a socket queue by the switching processes. Packets that are generated by processes on the router are queued directly to the correct interface's output queue.

The output of the show interface command in Cisco IOS Software shows both queues, which is confusing until you understand what the output is indicating.

```
router#show interface fastethernet 0/0
FastEthernet0/0 is up, line protocol is up
Hardware is DEC21140A, address is 0030.7b1d.2c00 (bia
0030.7b1d.2c00)
....
Queueing strategy: fifo
Output queue 0/40, 0 drops; input queue 0/75, 0 drops
```

Figure 1-32 illustrates the input and output queues in a different way that will help explain the output of the show interface command.

Figure 1-32. Input and Output Queues in Relation to the show interface Command
[View full size image]



Each process that switches or creates packets feeds the single output queue for each interface; this much is straightforward. If each process in Figure 1-32 has queued one packet on the interface shown, show interface would report four packets in output queue.

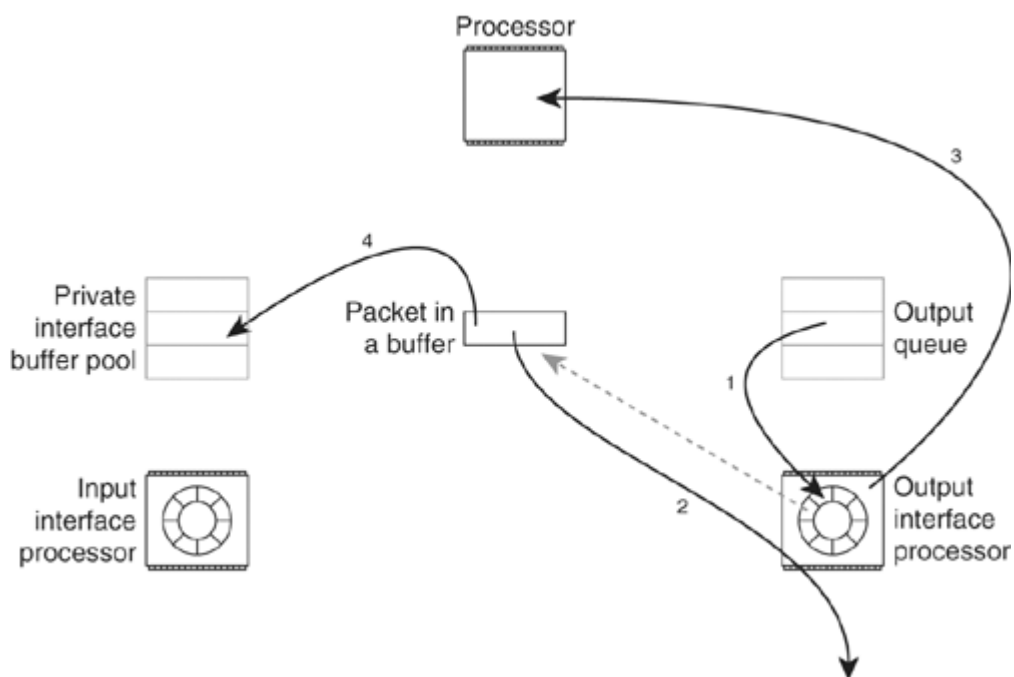
Each interface, however, feeds several input queues. The input queue count shown in the output of

show interface indicates the number of packets the interface has queued to processes that haven't been processed. Thus, the interface in Figure 1-32 has queued one packet toward process 1, three packets toward process 2, two packets toward process 3, and five packets toward process 4. Eleven packets have been queued by the interface and are waiting to be processed. The output of show interface would show the input queue for this interface with 11 packets.

Transmitting the Packet

The Layer 2 header, or MAC header, has now been rewritten, and the packet is shifted to the outbound interface. Figure 1-33 illustrates the transmission of the packet.

Figure 1-33. Transmitting the Packet



The following list describes the process of transmitting a packet:

1. If the packet is currently in the interface's output queue, it is moved to the transmit ring.
2. The packet is now transmitted by the interface processor from memory onto the network media.
3. The interface processor interrupts the processor to notify it of a completed packet transmission. This is the transmit interrupt. The code that runs during the transmit interrupt updates the interface counters.
4. During the transmit interrupt, the packet buffer is returned to the pool from which it was drawn when the packet was created (if it was locally generated) or the pool from which the packet buffer was drawn when it was placed on the inbound interface's receive ring. This completes the entire cycle, returning the resources to their original location.

After the packet buffer has been returned to the pool from which it was drawn and the interface counters have been updated, the transmit interrupt is released, and the processor continues running the process that was interrupted.

Hardware and Software show Commands

Figures 1-34 through 1-38 provide detailed information on the various show commands used to examine the hardware and software state of a Cisco router. Figure 1-34 explains the show memory command.

Figure 1-34. show memory Command

Memory pools	First address in the pool	Total memory in the pool	Used memory	Free memory	Lowest amount of free memory ever in this pool	Largest block of free memory
router#show memory	Head	Total (b)	Used (b)	Free (b)	Lowest (b)	Largest (b)
Processor	81905754	35629228	5710284	29918944	29879896	29455336
I/O	3B00000	5242880	2145016	3097864	3067088	3097820

Address	Bytes	Prev.	Next	Ref	PrevF	NextF	Alloc	PC	What
81905754	1460	0	81905D34	1			80384650		List Elements
81905D34	2960	81905754	819068F0	1			80384680		List Headers
819068F0	9000	81905D34	81908C44	1			8039FD00		Interrupt Stack
81908C44	44	819068F0	81908C9C	1			80EBD64C		*Init*
81908C9C	92	81908C44	81908D24	1			80EBD64C		BGP Router

Annotations for Figure 1-34:

- Memory pools: points to the first two rows of the first table.
- First address in the pool: points to the 'Head' column.
- Total memory in the pool: points to the 'Total (b)' column.
- Used memory: points to the 'Used (b)' column.
- Free memory: points to the 'Free (b)' column.
- Lowest amount of free memory ever in this pool: points to the 'Lowest (b)' column.
- Largest block of free memory: points to the 'Largest (b)' column.
- Start of block: points to the 'Address' column of the second table.
- Block size: points to the 'Bytes' column.
- Start of previous block: points to the 'Prev.' column.
- Start of next block: points to the 'Next' column.
- Number of processes referencing this block: points to the 'Ref' column.
- Previous free block: points to the 'PrevF' column.
- Next free block: points to the 'NextF' column.
- Indicates which process allocated this block: points to the 'Alloc' column.
- Process name: points to the 'What' column.

Figure 1-35 explains the show processes memory command.

Figure 1-35. show processes memory Command

```
router#show processes memory
```

Total: 35629228, Used: 5710380, Free: 29918848							
PID	TTY	Allocated	Freed	Holding	Getbufs	Retbufs	Process
0	0	74032	1808	3358588	0	0	*Init*
0	0	1116	275416	1116	0	0	*Sched*
0	0	1290760868	1283157048	1223504	164928	0	*Dead*
1	0	280	280	3864	0	0	Load Meter
2	0	483768	455956	14644	0	0	Exec
3	0	0	3936	6864	0	0	Check heaps
4	0	20248	0	27112	0	0	Chunk Manager
5	0	17108	0	6960	8316	0	Pool Manager
6	0	280	280	6864	0	0	Timers
7	0	280	280	6864	0	0	Serial Background

Annotations for Figure 1-35:

- Process identifier: points to the 'PID' column.
- Terminal line: points to the 'TTY' column.
- Number of bytes allocated by this process since it started running: points to the 'Allocated' column.
- Number of bytes freed since this process started running: points to the 'Freed' column.
- Number of bytes this process is currently holding: points to the 'Holding' column.
- Number of packet buffers taken by this process: points to the 'Getbufs' column.
- Number of packet buffers returned by this process: points to the 'Retbufs' column.
- Process name: points to the 'Process' column.

Figure 1-36 explains the show region command output.

Figure 1-36. show region Command

```

router#show region
Region Manager:
  Start      End          Size (b)    Class  Media  Name
0x07800000  0x07FFFFFF  8388608    Iomem  R/W    iomem2
0x20000000  0x21FFFFFF  33554432   Iomem  R/W    iomem
0x57800000  0x57FFFFFF  8388608    Iomem  R/W    iomem2:(iomem2_cwt)
0x60000000  0x67FFFFFF  125829120  Local  R/W    main
0x60008960  0x611682CB  18217324   IText  R/O    main:text
0x6116A000  0x61B4E17F  10371456   IData  R/W    main:data
0x61B4E180  0x61E2EEDF  3018080    IBss   R/W    main:bss
0x61E2EEE0  0x677FFFFF  94179616   Local  R/W    main:heap
0x70000000  0x71FFFFFF  33554432   Iomem  R/W    iomem:(iomem_cwt)
0x80000000  0x877FFFFF  125829120  Local  R/W    main:(main_k0)
0xA0000000  0xA77FFFFF  125829120  Local  R/W    main:(main_k1)
  
```

While it might appear that the number of bytes of memory allocated by a process minus the number of bytes of memory currently being held by a process should equal the number of bytes of memory the process is holding, this isn't true. For example, processes in Cisco IOS Software can allocate memory that is then given to another process, and processes can free memory that they did not allocate—and there are many other ways in which a process can end up owning memory that it's not using.

Figure 1-37 explains the show buffers command output.

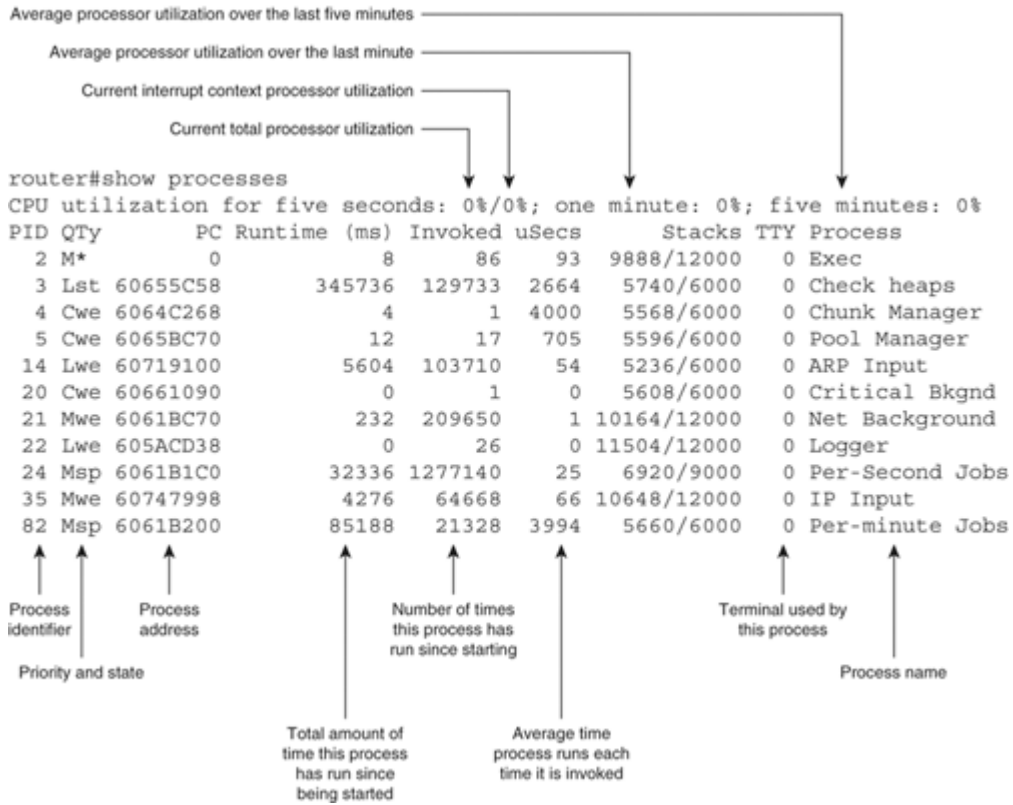
Figure 1-37. show buffers Command

```

Smallest number of buffers which will ever be in this pool
Maximum number of free buffers allowed in this pool
Number of buffers in this pool
Minimum number of free buffers allowed in this pool
Size of buffers in this pool
router#show buffers
....
Public buffer pools:
Small buffers, 104 bytes (total 5, permanent 5):
  2 in free list (2 min, 5 max allowed)
  3 hits, 0 misses, 0 trims, 0 created
  0 failures (0 no memory)
....
Number of buffer requests that could not be serviced
Number of buffer requests that have been serviced
Number of free buffers currently in the pool
Number of buffers created
Number of buffers freed
How many times the number of free buffers has gone below the minimum free
How many times a buffer could not be created because there was no memory
  
```

Figure 1-38 explains the show processes command.

Figure 1-38. show processes Command



This chapter includes the following topics:

- [Evolving packet-switching methods](#)
- [CEF tables](#)
- [CEF table entries](#)
- [Special adjacencies](#)
- [Switching a packet with CEF](#)
- [The CEF Epoch](#)
- [Configuring CEF/dCEF](#)

This chapter provides details about the architecture and methods of Cisco Express Forwarding, commonly referred to as CEF. The details contained in this chapter are an overview of CEF as it is applied to low-end Cisco platforms, such as the Cisco 1700, 2800, 3800, and 7200 series routers.

CEF and Multiprotocol Label Switching (MPLS) were recently rewritten in Cisco IOS Release 12.2S to provide an enhanced, more scalable, fully integrated switching path solution for Cisco routers and switches. The new pervasive switching path infrastructure is referred to as CEF/MFI (CEF/MPLS Forwarding Infrastructure).

This chapter focuses on the CEF implementation, prior to the CEF/MFI IOS Release 12.2S enhancement, while [Chapter 3](#), "CEF Enhanced Scalability," discusses CEF/MFI in detail. Network engineers must have an understanding of general packet switching and CEF to troubleshoot and maintain vastly growing and ever-changing networks.

Evolving Packet-Switching Methods

Packet switching is the method of moving a packet from a router's input interface to an output interface. When Cisco routers were first developed and deployed, only one way existed to switch packets; this was called process switching. As networks evolved, Cisco developed more scalable, cache-based packet-switching methods, known as fast switching and Cisco Express Forwarding.

Process Switching

Process switching refers to the method by which the router's CPU is directly involved in the forwarding decision. As a packet is process-switched through a router, its CPU is in charge of choosing the appropriate process to handle the packet and scheduling the running of the process.

The following list briefly outlines the order of operation of a packet as it is being process-switched through a router:

1. The router reads the packet off the wire connected to the inbound interface.
2. The router stores the packet in memory.
3. The router performs a lookup in the global routing table for the packet's destination IP address.
4. The router finds the next-hop IP address toward the destination by resolving any recursion within the routing table.
5. The router finds the physical Layer 2 header and outbound interface for the next-hop device. This information is written onto the packet to forward it to the next hop toward the destination; generally this involves looking in the Address Resolution Protocol (ARP) cache or some other table that maps a next-hop device to a Layer 2 address.
6. The router rewrites the physical Layer 2 header address onto the packet.

7. The router writes the packet from memory onto the wire connected to the outbound interface.

Early in the development of Cisco routers, it became apparent to the Cisco engineers that process switching was not going to be able to handle packets fast enough to attain the speeds needed; traffic flows were increasing at a rapid pace across networks, and the routers had to be designed to keep up.

A classic idea from computer science was applied to the problem: Why not cache the results of the IP next-hop and Layer 2 lookups and use them to switch the next packet toward a given destination? This concept resulted in fast switching.

Fast Switching

Fast switching relies on a demand-based cache to forward packets. The first packet is copied to packet memory, and the destination network or host is found in the fast-switching cache. The frame is then rewritten and sent to the outgoing interface that services the destination. If the destination address is not present in the fast-switching cache, the packet is returned to the process-switched path, where the processor attempts to build a cache entry which can be used to forward packets to the destination.

In late 1995, it became apparent that this demand-based caching system would not be able to keep up, especially on platforms such as the Cisco 12000 series routers, which rely heavily on distributed packet switching. The pace of change on the Internet backbone was increasing; this rapid change rate showed up as rapid increases in the size of routing tables on backbone routers.

Demand-based switching caches and rapid routing table changes do not work well together. Each time the cache becomes settled, a new change takes place in the routing table, necessitating clearing a portion of the cache. Each time a section of the cache is removed, a number of packets are handed off to the process-switched path to be processed and then forwarded. If the rate of change becomes great enough, the percentage of packets switched through the demand-based cache becomes so low that little or no efficiency is gained by the caching mechanism. This, in fact, is what the Cisco engineers were experiencing.

You can find more information on process switching and fast switching in *Inside Cisco IOS Software Architecture* (Cisco Press, 2000).

By March 1996, a new mechanism was designed that would overcome these problems with the fast cache—Cisco Express Forwarding (CEF).

What Is CEF?

CEF switching is a proprietary, advanced Layer 3 IP switching mechanism that was designed to tackle the deficiencies associated with fast-switching demand-caching mechanisms. CEF optimizes performance, scalability, and resiliency for large-scale, complex networks with dynamic traffic patterns. CEF's retrieval and forwarding technique is less CPU intensive than process or fast switching. This results in higher throughput when CEF is enabled. CEF also provides a common forwarding code base that facilitates the development and maintainability of features performed in the forwarding path.

Since 1999, CEF has been the default switching path on most Cisco router platforms that implement software-based switching, and it is the mechanism used by virtually every Cisco router that implements hardware-based switching. This chapter focuses on the understanding and operation of CEF software-based switching.

Distributed CEF (dCEF) switching is a mechanism that offloads the packet-forwarding process from the router's CPU to the CPU on a Versatile Interface Processor (VIP) or line card installed in a distributed platform, such as the Cisco 7500 and Cisco 12000 series routers. In a distributed platform, each VIP or line card maintains a copy of the CEF forwarding data structures, so the switching is performed at the distributed level. Because the packet switching is performed at the distributed layer, the router's CPU can have significantly more resources to perform routing, network management, and other services.

A table that can be used to switch packets needs to store the following information:

- The output interface

- The Layer 2 header rewrite string

The output interface is required for the router to know the specific interface on which a packet will be transmitted outbound, toward a destination. The Layer 2 header rewrite string contains the entire header needed to encapsulate the packet and forward it to the directly connected next-hop device. If the next-hop device is reachable over an Ethernet interface, the MAC header rewrite string would consist of the entire Ethernet header, including the destination MAC address and all the fields required in the Ethernet header. CEF's implementation maintains this information in two tables, which are described in the next sections.

CEF Tables

The two main components of CEF are as follows:

- The Forwarding Information Base (FIB)
- The adjacency table

Forwarding Information Base (FIB)

The FIB is a dynamically built database of information used by the router to make prefix-based packet-forwarding decisions. The information stored in the FIB is a collection of necessary details copied from the router's global routing table, as well as host route entries. The FIB also precomputes recursive path lookups. The FIB is organized in a way to optimize fast retrieval and longest-match lookups. The FIB is a multiway trie, or simply an mtrie, that stores reachability information about destinations and their respective next hops, which the router uses to reach each destination.

Note

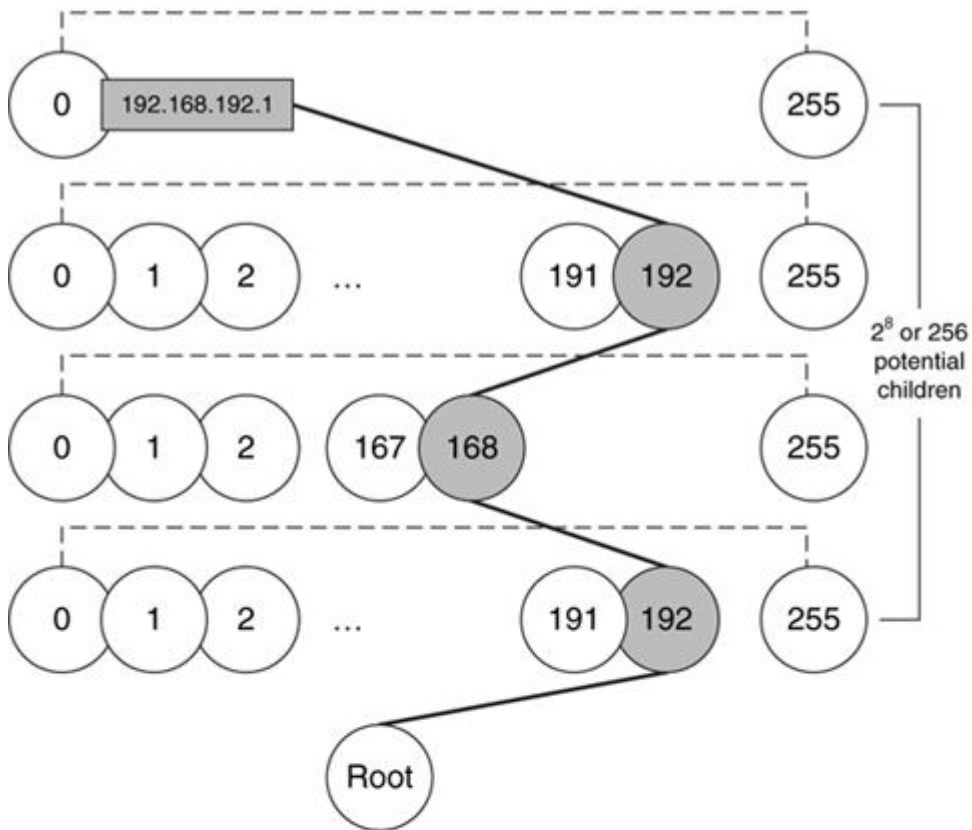
An mtrie is a searchable data structure, made up of nodes and leaves, that focuses on fast retrieval of data, rather than the primary concern of sorting and storing information.

In the FIB, each node has 256 children that are pointers to lower levels in the data structure. The FIB starts with a default root node, representing the starting point to search for a node representing a given prefix. The stride length of an mtrie is the number of bits represented by each level of the mtrie. CEF's mtrie implementation supports variable mtrie stride patterns, allowing increased lookup speed and optimized memory utilization. The mtrie stride pattern varies by platform.

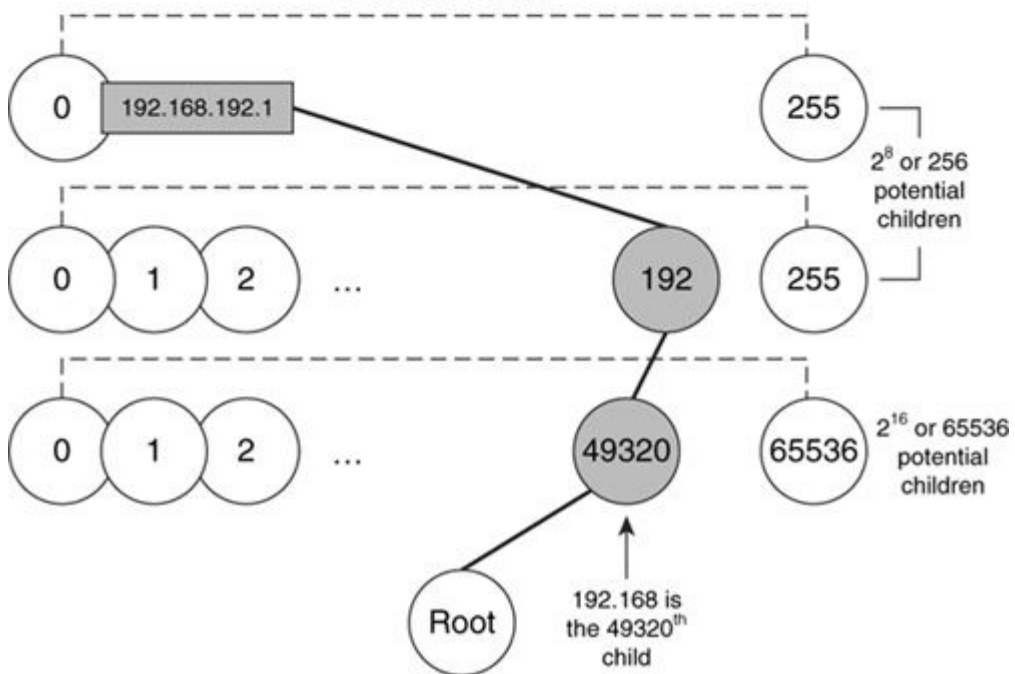
For the 8-8-8-8 mtrie stride pattern, the maximum number of levels is 4, but the trie will prune the sublevels if all the child pointers of a node refer to the same object. For example, an empty mtrie consists only of the root trie node, and all the pointers in that node point to "NULL." With an 8-8-8-8 stride pattern and a destination of 192.168.1.47, the first lookup would find 192, the second would find 168, the third would find 1, and the fourth would find 47.

Each trie node pointer can point to either another node or a leaf. A leaf is a terminating trie data structure that is embedded in the data object that the trie's application stores. Each data object can only be stored once in the trie. Multiple mtrie nodes can point to the same trie leaf multiple times. [Figure 2-1](#) illustrates an 8-8-8-8 stride pattern.

Figure 2-1. Two Different FIB Structures
8-8-8-8 mtrie



16-8-8 mtrie



The FIB is often thought of as a pure software structure with three levels of nodes, with each level having children. While this is how the FIB is built in software on the main processor of every router running Cisco IOS

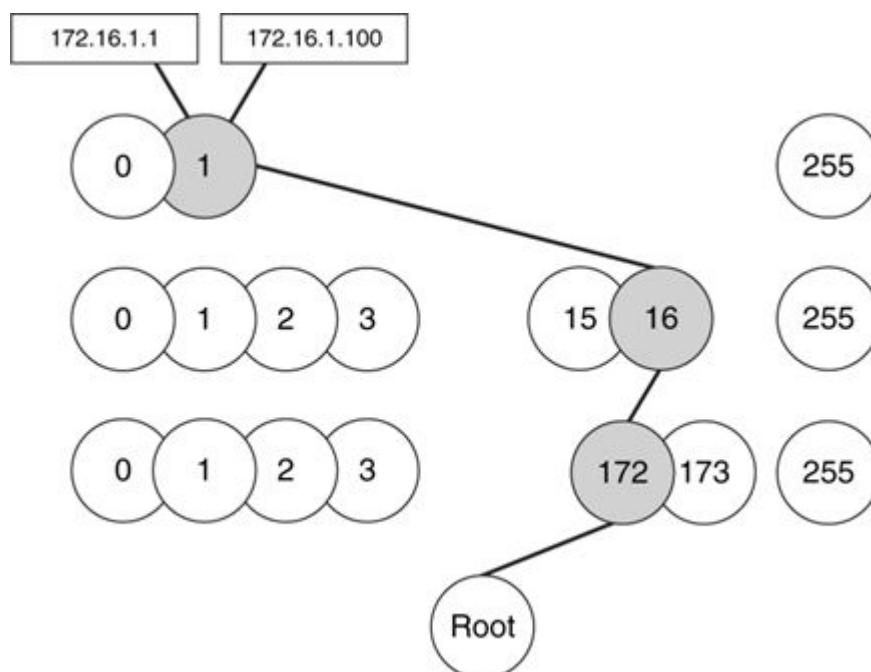
Two modal points exist within the prefix distribution graph. About 50 percent of the prefixes have mask lengths of less than 16 bits, while the remaining 50 percent have mask lengths of 24 bits. This implies that the FIB used on an Internet backbone needs to perform well at those two prefix lengths.

The Cisco 12000 series router uses a 16-8-8 stride pattern for this reason. About 25 percent of the lookups can take place with one lookup in the FIB, while most of the remaining 75 percent require only two lookups. In reality, the traffic spread makes this split about 50 percent to 50 percent, because a large amount of the traffic switched on the Internet backbone is destined to one of the 16-bit or shorter destinations in the routing table.

To summarize, there are an infinite number of ways to build an FIB, but the bottom line is that they scale differently for different applications. The stride pattern is a trade-off between memory usage and lookup performance. The best lookup performance could be achieved with a stride pattern of 32, but this approach is not scalable, because it would consume too much memory.

In [Figure 2-4](#), the path taken through the table to find the prefix 172.16.1.1/32 is highlighted. From the root, the 172nd node is chosen. This node points to the 16th node, which in turn, points to the first node. Finally, the first node points to the FIB mtrie leaf structure, 172.16.1.1. The same path along the nodes is used when looking up the FIB mtrie leaf entry for 17.16.1.100/32.

Figure 2-4. Structure of the FIB



If the final node exists, you know that the network in question is reachable. The behavior in this example is called a longest-match lookup, which the router performs on the destination IP address of packets being forwarded through the router. The output of the command `show ip cef [[network]][mask]` displays longest-match lookups for entries in the FIB. On the contrary, exact-match lookups are performed by the router for packets destined to the router's control plane. [Example 2-1](#) shows a longest-match lookup for a prefix in the FIB.

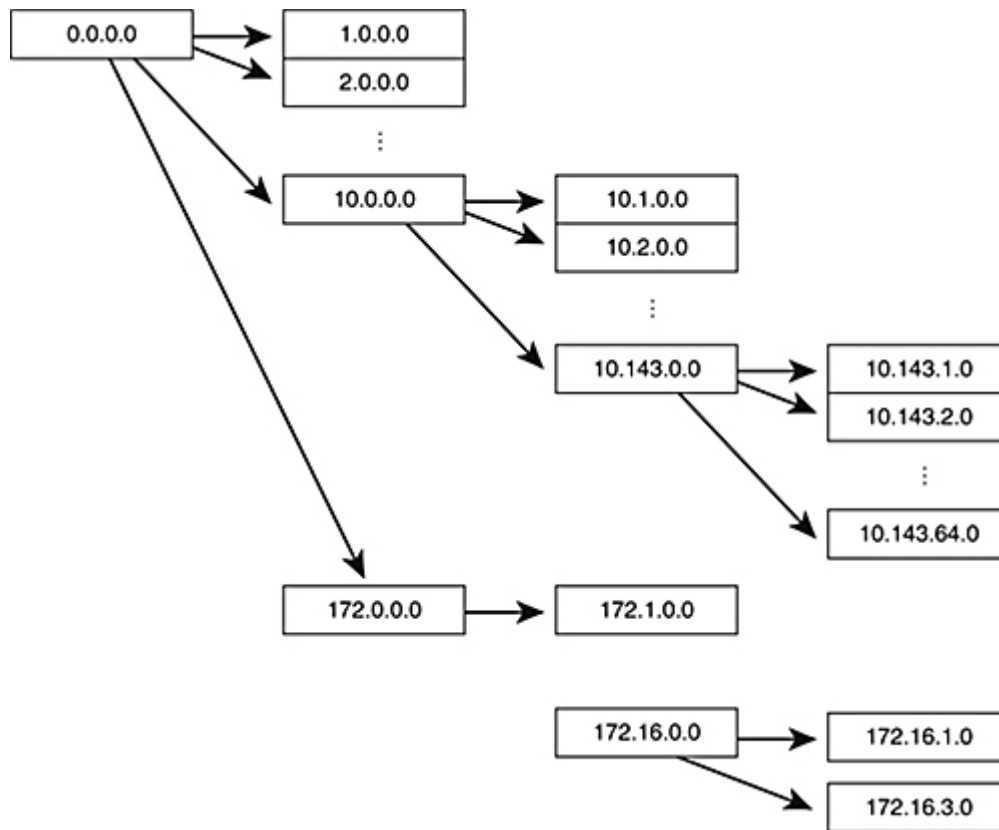
Example 2-1. show ip cef Command

```
Router#show ip cef 192.168.1.0 255.255.255.0
192.168.1.0/24, version 59, epoch 0, cached adjacency 10.0.1.2
0 packets, 0 bytes
  via 10.0.1.2, GigabitEthernet0/1, 0 dependencies
    next hop 10.0.1.2, GigabitEthernet0/1
    valid cached adjacency
```

You still don't know what the next hop of the prefix is or which Layer 2 header rewrite string should be used to forward the packet to the next hop.

The FIB corresponds directly with the information stored in the routing table, as [Example 2-2](#) and [Figure 2-5](#) illustrate.

Figure 2-5. The FIB's Relationship to the Routing Table



Example 2-2. show ip route Command

```
Router#show ip route
 10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D EX   10.143.0.0/18 [170/2170112] via 208.0.11.11, 2w1d, Serial0/2
D EX   10.143.64.0/18 [170/2170112] via 208.0.11.11, 2w2d, Serial0/2
C      172.16.1.0/24 is directly connected, FastEthernet0/0
C      172.16.3.0/24 is directly connected, Serial0/1
```

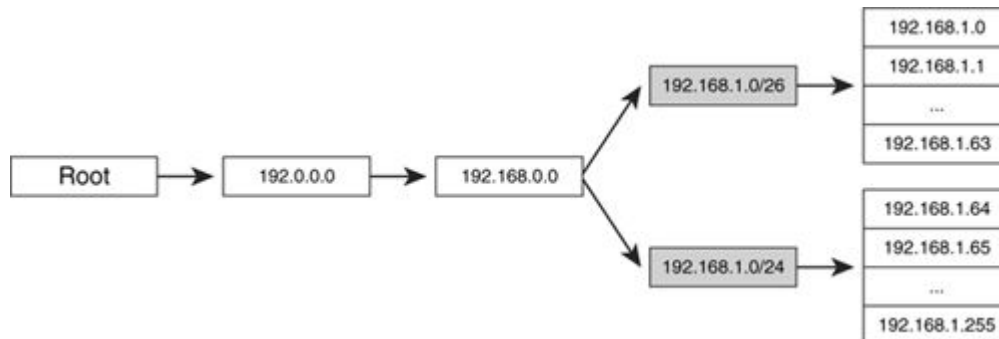
Each entry in the routing table relates to zero or one FIB entry. The FIB is built and maintained in the same manner as the routing table. Any update to an entry in the routing table triggers an update to the respective prefix or prefixes in the FIB. For example, if the next-hop IP address changes for a route in the routing table, the FIB entry's next-hop IP address and, if necessary, the respective output interface are updated.

Updates can occur whether or not packets are being switched to each destination in the routing table. In fact, this update can occur before any packets are switched to a destination within the routing table.

One question that arises when considering the design of switching tables is how the data structures handle overlapping prefixes. How would the FIB install two prefixes, or destinations—say 192.168.1.0/24 and 192.168.1.0/26—that overlap? CEF would build two FIB entries for these prefixes, as illustrated in Figure 2-6.

Figure 2-6. Overlapping Prefixes or Destinations

[\[View full size image\]](#)



The FIB would be built with two entries at the last level to represent 192.168.1.0/24 and 192.168.1.0/26. One entry for 192.168.1.0/26 is built to represent the 192.168.1.0/26 prefix, pointing to 0–63 children. Another entry is built for 192.168.1.0/24, pointing to the rest of the 64–255 children within this address space. CEF manages overlapping prefixes by pointing the correct blocks of node pointers to the FIB entries.

This allows the FIB to continue to be directly related to the routing table and, at the same time, provides a simple way of handling overlapping prefixes without any other special information that would need to be consulted during the switching of a packet.

The Adjacency Table

After the router has determined a destination's reachability and its next hop, the next step is to discover the outbound interface, the Layer 2 header rewrite string, and any other information that might be needed to transmit a packet toward the next hop. This information is stored in the adjacency table.

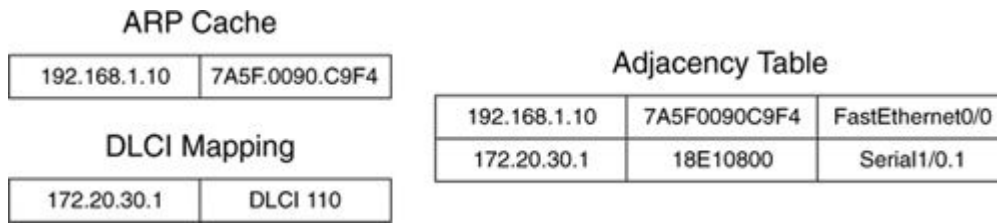
Nodes in a network are adjacent if they can reach each other within a single hop across a link. The adjacency table contains information about each directly connected host, router, or other device.

Adjacencies are discovered and added to the adjacency table through manual configuration, or they are dynamically learned through the ARP mechanism. Adjacencies can also be learned from point-to-point or multipoint interface configuration. A point-to-point interface uses a single adjacency for the interface, because point-to-point implies only one attached destination is through the interface. A multipoint interface can have multiple attached destinations, so it uses a unique Layer 2 rewrite string for each host IP address learned on that interface and completes the adjacency using the information from ATM or Frame Relay static mappings or inverse ARP. Adjacencies are selected as a result of a lookup on the combination of the next-hop IP address, output interface, and connection ID, if necessary. Connection IDs are used to decipher between nondistinct next-hop IP address and output interface pairs that can exist on ATM or multipoint interfaces.

The ARP cache and other mapping tables of this type are directly related to the adjacency table. Figure 2-7 illustrates this relationship.

Figure 2-7. Adjacency Table's Relationship to Other Address-Mapping Tables

[\[View full size image\]](#)

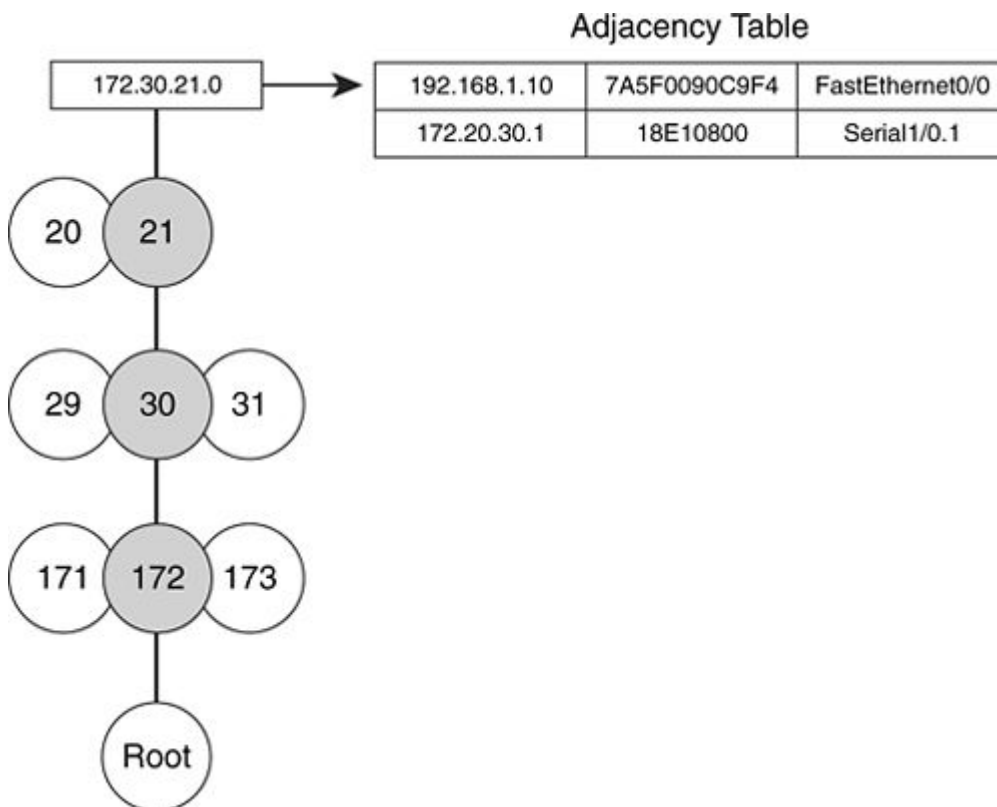


In [Figure 2-7](#), the destination MAC address is embedded in a complete Layer 2 rewrite header for the media associated with the outbound interface and mapped to the next-hop IP address in the adjacency table. To rewrite the Layer 2 rewrite header for any given next hop reachable from this router, simply locate the correct next hop in the adjacency table, replace the current Layer 2 header on the packet with the precomputed Layer 2 header rewrite string, and transmit the packet on the correct interface.

Relating the CEF Tables

Now that you have a better understanding of the FIB and adjacency tables, this section describes how they are related. [Figure 2-8](#) illustrates the relationship.

Figure 2-8. Relating the FIB and Adjacency Table



The final node in the FIB points to the correct entry in the adjacency table. If the routing table changes so that the next hop of a prefix is different, this pointer is simply moved to the new adjacency table entry.

CEF Table Entries

A FIB entry is a leaf data structure in the mtrie that contains the following items:

- Mtrie leaf data structure— This data structure includes the prefix IP address, subnet mask length, and refcount. The refcount is the number of leaves referenced by other nodes in the mtrie.
- Attributes— Attributes associated with FIB entries include the type flags (that is, RECEIVE, ATTACHED), which are described in detail in the next section. The quality of service (QoS) group information for a specific prefix is included in the attributes, as well as the traffic index. The traffic index is an accounting bucket used to count the number of packets that reference the prefix.
- One or more FIB paths— The FIB paths provide specific directions to a prefix and pointers to the associated adjacency.
- Load balance structures— If multiple paths exist to a prefix, the FIB entry will include pointers to loadinfo structures.

FIB Entries

The FIB includes the following entries, referred to by the FIB as flags:

- Attached
- Connected
- Receive
- Recursive
- Default Route Handler
- ADJFIB
- Learned from IGPs

The following sections discuss each of these entries, detailing how CEF uses them. FIB entries can be viewed on a router using the `show ip cef` command, as shown in the examples provided in the following sections.

Attached FIB Entry

An attached FIB entry is created when the destination is directly attached to the router. In [Example 2-3](#), the attached FIB entry represents a statically configured route with the next hop pointing to an interface attached to the router.

Example 2-3. Attached FIB Entry

```
Router#show ip route 10.1.1.0 255.255.255.0
Routing entry for 10.1.1.0/24
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
  * directly connected, via Serial0/1
    Route metric is 0, traffic share count is 1

Router#show ip cef 10.1.1.0 255.255.255.0
10.1.1.0/24, version 56, attached, cached adjacency to Serial0/1
0 packets, 0 bytes
  via Serial0/1, 0 dependencies
  valid cached adjacency
```

Connected FIB Entry

A connected FIB entry is an attached FIB entry that is created as a result of an IP address being configured on a router's interface. In [Example 2-4](#), note that the entry shows as being attached as well as connected, while static routes configured with an interface as their next hop are only flagged as attached.

Example 2-4. Connected FIB Entry

Code View: [Scroll](#) / [Show All](#)

```
Router#show ip route 192.168.30.0 255.255.255.0
Routing entry for 192.168.30.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
  * directly connected, via Serial0/1
    Route metric is 0, traffic share count is 1

Router#show ip cef 192.168.30.0 255.255.255.0
192.168.30.0/24, version 50, attached, connected, cached adjacency to
Serial0/1
0 packets, 0 bytes
  via Serial0/1, 0 dependencies
  valid cached adjacency
```

Receive FIB Entry

CEF uses a receive FIB entry when the IP address is configured on one of the router's interfaces. The router must accept and process packets destined to receive entry, 172.17.1.12, rather than switch the packets through the router out another interface. In [Example 2-5](#), note that the FIB entry is flagged as receive because it is the IP address assigned to the interface.

Example 2-5. Receive FIB Entry

```
Router#show running-configuration interface Serial0/3
interface Serial0/3
 ip address 172.17.1.12 255.255.255.0

Router#show ip cef 172.17.1.12 255.255.255.255
172.17.1.12/32, version 17, epoch 0, receive
```

The following list describes two special receive entries:

- 172.17.1.0/32— A receive FIB entry is built for the all-0s host address of a network. This entry allows the router to receive broadcasts sent to the all-0s host address on the network segment.
- 172.17.1.255/32— A receive FIB entry is built for the all-1s host address of a network. This entry allows the router to receive broadcasts sent to the all-1s host address on the network segment.

Recursive FIB Entry

The concept of performing a lookup for the next-hop IP address toward a destination is called a recursive lookup. Recursive routes are routes that require the forwarding and adjacency information of another route. A recursive FIB entry is used when the router must look up the next-hop IP address of a prefix referenced to find the corresponding Layer 2 header rewrite string and output interface for the destination prefix. Border Gateway

Protocol (BGP) is the most common source of recursive entries, because the IP addresses used as the next hop for advertised prefixes are almost never directly connected to the router. In [Example 2-6](#), note the FIB entry shows as recursive.

Example 2-6. Recursive FIB Entry

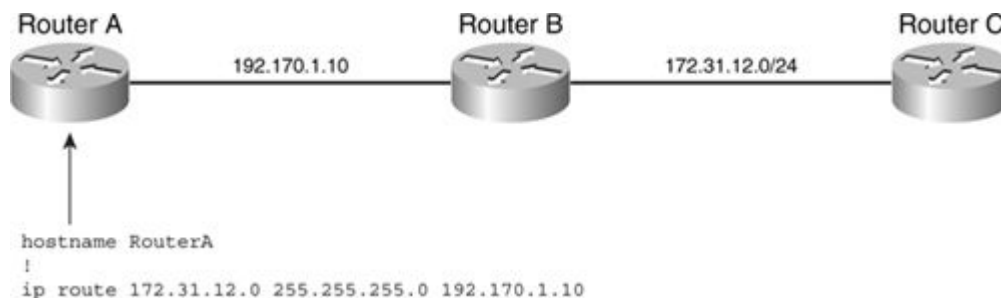
```
Router#show ip route 172.31.12.0 255.255.255.0
Routing entry for 172.31.12.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 192.170.1.10
    Route metric is 0, traffic share count is 1

Router#show ip cef 172.31.12.0 255.255.255.0
172.31.12.0/24, version 55, cached adjacency 192.170.1.10
0 packets, 0 bytes
  tag information set
    local tag: 4
  via 192.170.1.10, 0 dependencies, recursive
    next hop 192.170.1.10, FastEthernet0/1 via 192.170.1.10/32
    valid cached adjacency
```

[Figure 2-9](#) illustrates recursive routing.

Figure 2-9. Recursive Routing

[\[View full size image\]](#)



In [Figure 2-9](#), Router A's only knowledge of the 172.31.12.0/24 network is through the static route configured. The router installs this static route with a next-hop IP address of 192.170.1.10. When switching a packet toward a destination on the 172.31.12.0/24 network, Router A must perform another lookup on 192.170.1.10 to determine the respective outbound interface, Layer 2 header rewrite string, and other next-hop information installed for that destination.

Internet backbone routers carry large routing tables that are mostly comprised of BGP routes. The majority of BGP routes are recursive routes. For example, most BGP peering is performed between loopback interfaces. The next-hop IP address of received prefixes is generally the peer's loopback address, which is not connected to the local router. Therefore, the router must perform multiple route lookups on a prefix's next-hop IP address for a single destination, so performance becomes a factor. To reduce performance impact, CEF performs recursive resolution.

CEF performs recursive resolution by following the recursive chain of lookups down to the leaf level. The FIB entry is flagged as recursive and is populated with the forwarding details from the path of the next hop, also referred to as the parent. Recursive routes are resolved when a recursive path is created. If a valid next hop is not found upon initial creation, the recursive path is queued in a list for periodic reresolution by the CEF background resolution process. CEF maintains a dependency list to keep track of changes. All the children paths depending

on a parent's path are installed in the parent's dependents list. If a parent's path information changes, any dependents will also be reresolved.

CEF does not limit the amount of levels required for recursion. However, multilevel recursion used for load balancing was not supported until the CEF/MFI rewrite in Cisco IOS Software Release 12.2S, which is covered in [Chapter 3](#). On a router running pre-MFI IOS, multilevel recursion with load balancing results in CEF using all the paths at the first level and then choosing only one of the paths at the subsequent levels. The results do not guarantee that load balancing will occur because of lack of code support.

Default Route Handler FIB Entry

The FIB installs the default route handler entry as a default entry placeholder. Packets destined to nonspecific destinations installed in the FIB use this entry and are discarded. Any default routes that are statically configured or learned from dynamic routing protocols are installed in place of this entry. However, if the default route is removed from the routing table, the FIB default route handler entry is reinstalled as a placeholder. [Example 2-7](#) provides an example of the default route handler FIB entry.

Example 2-7. Default Route Handler FIB Entry

```
Router#show ip cef 0.0.0.0 0.0.0.0
0.0.0.0/0, version 2, epoch 0, attached, default route handler
0 packets, 0 bytes
  via 0.0.0.0, 0 dependencies
  valid no route adjacency
```

ADJFIB FIB Entry

The ADJFIB entry is a special FIB entry that reflects an entry in the adjacency table. The best way to describe this entry is by providing an example. For a broadcast network, the FIB table will represent the connected network entry as 1.1.1.0/24. This entry will point to a glean adjacency, indicating that packets matching host prefixes within the connected network entry, such as 1.1.1.1, will need to glean more information to forward the packet. On an Ethernet interface, the router uses ARP to learn the necessary details used to forward packets destined to these host entries. Upon receiving an ARP response, the router installs the Layer 2 rewrite string and output interface details in the ARP table and the adjacency table. The router then installs an ADJFIB entry in the FIB table to reflect this adjacency. [Example 2-8](#) shows a representation of an ADJFIB FIB entry.

Example 2-8. ADJFIB FIB Entry

```
interface Ethernet0/0
 ip address 1.1.1.2 255.255.255.0

Router#show ip cef 1.1.1.0 255.255.255.0
1.1.1.0/24, version 4, epoch 0, attached, connected
0 packets, 0 bytes
  via Ethernet0/0, 0 dependencies
  valid glean adjacency

Router#show ip cef 1.1.1.1
1.1.1.1/32, version 10, epoch 0, connected, cached adjacency 1.1.1.1
0 packets, 0 bytes
  via 1.1.1.1, Ethernet0/0, 0 dependencies
  next hop 1.1.1.1, Ethernet0/0
  valid cached adjacency
```

Learned from IGPs

Entries learned from IGPs are the most common source of prefixes installed in the FIB. [Example 2-9](#) illustrates a route learned through Open Shortest Path First (OSPF), which is programmed in the FIB table as a valid cached adjacency.

Example 2-9. FIB Entry Learned Through an IGP

Code View: [Scroll](#) / [Show All](#)

```
Router#show ip route 100.1.1.0
Routing entry for 100.1.1.0/24
  Known via "ospf 1", distance 110, metric 10, type extern 2, forward
metric 10
  Last update from 38.0.0.2 on Ethernet1/0, 00:03:11 ago
  Routing Descriptor Blocks:
  * 38.0.0.2, from 46.0.0.2, 00:03:11 ago, via Ethernet1/0
    Route metric is 10, traffic share count is 1

Router#show ip cef 100.1.1.0
100.1.1.0/24, version 17, epoch 0, cached adjacency 38.0.0.2
0 packets, 0 bytes
  via 38.0.0.2, Ethernet1/0, 0 dependencies
  next hop 38.0.0.2, Ethernet1/0
  valid cached adjacency
```

A set of generic FIB entries are also created on every router; these entries are covered in the next section.

Generic FIB Entries

Any router can have, at the minimum, the FIB entries listed in [Example 2-10](#), which are not tied to any particular interface. Note that the Interface column for these entries is empty.

Example 2-10. Generic FIB Entries

```
Router#show ip cef
Prefix           Next Hop           Interface
0.0.0.0/32       receive
224.0.0.0/4      drop
224.0.0.0/24     receive
255.255.255.255/32 receive
```

The following list describes each of the entries shown in [Example 2-10](#):

- 0.0.0.0/32 receive— This entry allows the router to receive packets transmitted to the 0.0.0.0 broadcast address. Because Routing Information Protocol (RIP) and other protocols, such as DHCP/BOOTP, use broadcast packets to transmit information, the router must receive packets transmitted to the broadcast address.
- 224.0.0.0/4 drop— This entry drops packets destined to this reserved IP multicast address space when IP Multicast Routing is disabled. If IP Multicast Routing is enabled, the FIB entry will be reported as a punt adjacency. Note that CEF does not support the switching of multicast packets. By default, multicast packets are fast-switched through a router.

- 224.0.0.0/24 receive— The range of multicast IP addresses, 224.0.0.0 through 224.0.0.255, is reserved for the use of routing and discovery protocols. For example, Enhanced Interior Gateway Routing Protocol (EIGRP) transmits hello packets to multicast address 224.0.0.10 to maintain neighbor relationships. This FIB entry allows the router to receive and process packets destined to this IP address range.
- 255.255.255.255/32 receive— This entry allows the router to receive packets transmitted to the link local broadcast address.

Interface-Specific FIB Entries

The three types of interfaces of primary concern when looking at a normal FIB table are as follows:

- Multiaccess interfaces (including broadcast, such as Ethernet)
- Point-to-point interfaces (including most WAN interfaces, such as Packet over SONET [PoS], Frame Relay, and High-Level Data Link Control [HDLC] links)
- Links that are configured to use 31-bit subnet masks

The next sections describe each of these FIB entries.

FIB Entries Built for a Multiaccess Network Interface

A set of FIB entries is built for a multiaccess (or broadcast) interface by default. [Example 2-11](#) shows the set of FIB entries built for an Ethernet interface.

Example 2-11. FIB Entries Built for a Multiaccess Network Interface

```
Router#show ip cef
Prefix          Next Hop          Interface
172.16.12.0/24  attached         Ethernet3/0
172.16.12.0/32  receive
172.16.12.3/32  receive
172.16.12.255/32 receive
```

The following list describes the FIB entries associated with a multiaccess network interface:

- 172.16.12.0/24— This connected FIB entry is built for the network. The subnet prefix points to a glean adjacency. A glean adjacency exists when a specific next hop is directly connected, but no Layer 2 header rewrite string is associated with the entry. Glean adjacencies are covered in the next section.
- 172.16.12.0/32— This receive FIB entry is built for the all-0s host address on the network segment.
- 172.16.12.3/32— This receive FIB entry is built for the local interface address so that the router can receive and process packets sent to this IP address.
- 172.16.12.255/32— This receive FIB entry is built for the all-1s host address on the network segment.

FIB Entries Built on a Point-to-Point Network Interface

A set of FIB table entries are built for point-to-point interfaces. The primary difference between these entries and FIB entries built for a multiaccess interface is that the 192.168.6.0/24 attached FIB entry points to a point-to-point adjacency for the interface, as shown in [Example 2-12](#).

Example 2-12. FIB Entries Built on a Point-to-Point Network Interface

```
Router#show ip cef
```

Prefix	Next Hop	Interface
192.168.6.0/24	attached	Serial5/3
192.168.6.0/32	receive	
192.168.6.3/32	receive	
192.168.6.255/32	receive	

FIB Entries Built on a 31-Bit Prefix Network Interface

RFC 3021 permits the use of a 31-bit prefix length on point-to-point links to conserve IP address space. Some FIB entries are not built on these links because no broadcast addresses are on a link with only two hosts. As shown in [Example 2-13](#), an interface configured with a 31-bit mask can only have the connected FIB entry for the network and the receive FIB entry for the locally assigned IP address.

Example 2-13. FIB Entries Built on a 31-Bit Prefix Network Interface

```
Router#show ip cef
```

Prefix	Next Hop	Interface
10.0.6.2/31	attached	Serial5/3
10.0.6.3/32	receive	

Special Adjacencies

In addition to entries in the FIB, the adjacency table contains special types of adjacencies, including the following:

- Auto
- Punt
- Glean
- Drop
- Discard
- Null
- No route
- Cached and uncached
- Unresolved

The following sections describe each type of adjacency in detail.

Auto Adjacencies

Auto adjacencies are the most common type of adjacencies. [Example 2-14](#) illustrates an auto adjacency installed for interface Serial2/0.

Example 2-14. Auto Adjacencies

```
interface Serial2/0
 ip address 10.1.1.1 255.255.255.252

Router#show adjacency internal
Protocol Interface          Address
IP          Serial2/0        point2point(5)
                                0 packets, 0 bytes
                                0F000800
                                CEF   expires: 00:02:39
                                refresh: 00:00:39
                                Epoch: 0
                                Fast adjacency disabled
                                IP redirect enabled
                                IP mtu 1500 (0x0)
                                Fixup disabled
                                Adjacency pointer 0x1BC6D48, refCount 5
                                Connection Id 0x000000
                                Bucket 10
```

The protocol type (in this case IP), the Layer 2 header rewrite string, the source of the adjacency entry, and the outbound interface are all included in an adjacency entry. The next hop can be an actual address or it can be marked as point-to-point, which indicates that the next hop is reachable over a point-to-point interface.

Punt Adjacency

Punt adjacencies are used when packets to a destination cannot be CEF switched. When a feature is not supported in the CEF switching path, the punt adjacency allows a packet to be switched using the next slower switching mechanism configured on the router. Packets that are punted from the CEF path are sent up to the fast-switching or process-switching path to be handled. All FIB entries that point to a punt adjacency can be viewed using the Cisco IOS command `show ip cef adjacency punt`. [Example 2-15](#) shows a punt adjacency installed for a multicast address on a router enabled for IP Multicast Routing. IP multicast packets are not supported in the CEF switched path; they are fast- or hardware-switched through a router, depending on the platform type.

Example 2-15. Punt Adjacency

```
Router#show ip cef 239.1.1.1
224.0.0.0/4, version 17, epoch 0
0 packets, 0 bytes
  via 0.0.0.0, 0 dependencies
    next hop 0.0.0.0
    valid punt adjacency
```

Glean Adjacency

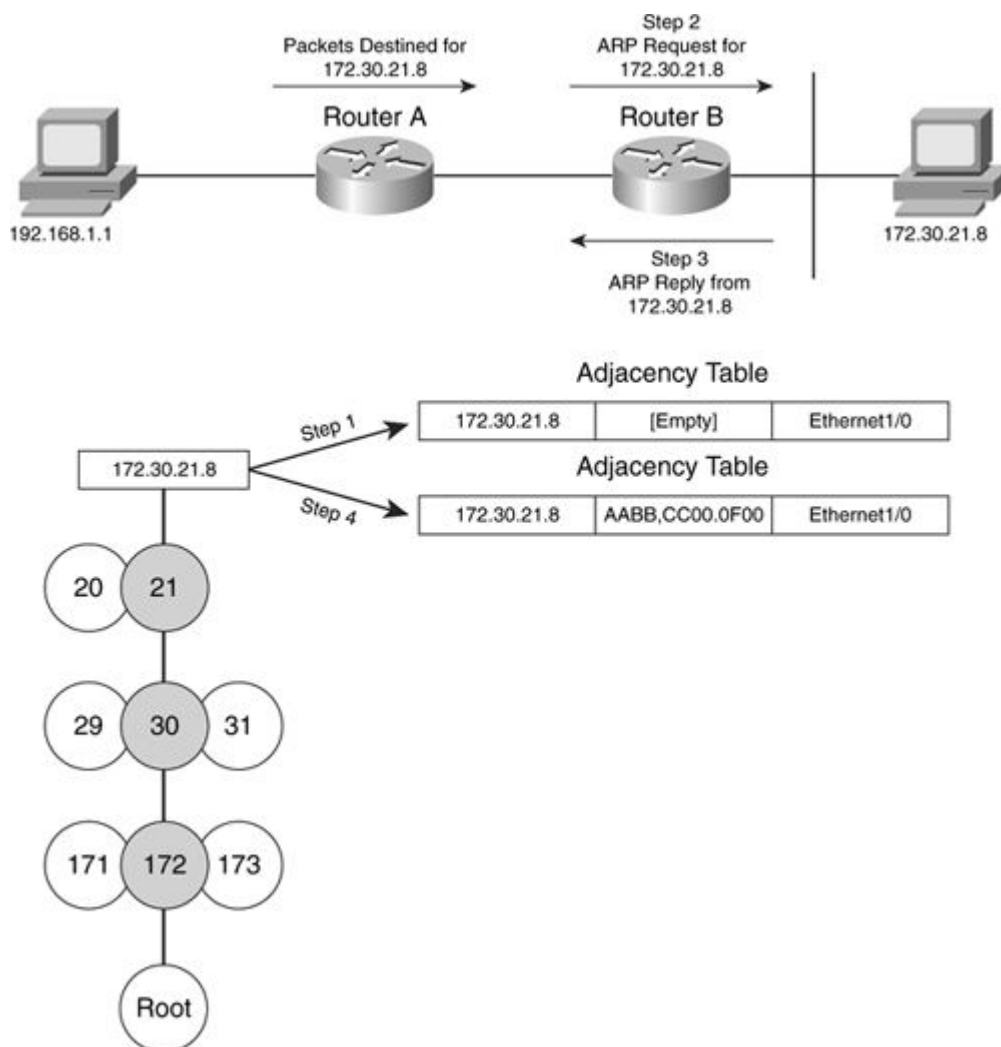
When a router is directly connected to a segment shared by multiple hosts, the FIB table on the router maintains a prefix for the subnet, rather than an individual prefix for each host. The subnet prefix points to a glean adjacency. When packets need to be forwarded to a specific host, the adjacency database is gleaned for the specific prefix. The FIB entry pointing to a glean adjacency indicates that any addresses within this range should be directly connected to the router, but no Layer 2 header rewrite string is associated with the specific host.

All FIB entries that point to a glean adjacency can be seen using the Cisco IOS command `show ip cef adjacency glean`. The following list and [Figure 2-10](#) explain a simple glean operation:

1. The first packet arrives at Router B from 192.168.1.1 through Router A. Router B performs a FIB lookup for the destination IP address, follows the FIB table down to the longest match, and finds the 172.30.21.0/24 entry. This entry points to a glean adjacency. A glean adjacency causes CEF to punt the packet up to the process level, which in turn hands the packet off to ARP to resolve the Layer 2 header rewrite information for the destination IP address.
2. Router B sends an ARP request for this host destination. The host, 172.30.21.8, responds to the ARP request with its MAC address embedded.
3. Router B receives the ARP reply and builds a new ARP entry.
4. Upon creation of the adjacency, a new FIB ADJFIB entry is created for this host, 172.30.21.8/32, on Router B. Now packets switched through Router B destined for this host IP address can follow the FIB to this new host entry and then to the adjacency table entry, which contains the MAC layer header rewrite string for 172.30.21.8.

Figure 2-10. Glean Adjacency

[\[View full size image\]](#)



Drop Adjacency

The FIB points to a drop adjacency when packets switched to this destination should be dropped upon being received. Drop adjacencies obey any configured Internet Control Message Protocol (ICMP) requirements.

[Example 2-16](#) shows a drop adjacency installed for multicast range 224.0.0.0/4 on a router that does not have IP Multicast Routing enabled.

Example 2-16. Drop Adjacency

```
Router#show ip cef 224.0.1.1
224.0.0.0/4, version 19, epoch 0
0 packets, 0 bytes
  via 0.0.0.0, 0 dependencies
    next hop 0.0.0.0
    valid drop adjacency
```

Discard Adjacency

Discard adjacencies are used for addresses that are part of a loopback interface's subnet, but are not actually configured on the loopback interface. In [Example 2-17](#), IP addresses 192.168.1.2 through 192.168.1.254 would all point to a discard adjacency, and packets received by the router, destined to those addresses, would be silently discarded. Packets dropped by a discard adjacency do not generate ICMP messages by default.

Example 2-17. Discard Adjacency

```
interface loopback0
 ip address 192.168.1.1 255.255.255.0

Router#show ip cef 192.168.1.2
192.168.1.0/24, version 18, epoch 0, attached, connected
0 packets, 0 bytes
  via Loopback0, 0 dependencies
    valid discard adjacency
```

All FIB entries pointing to a discard adjacency can be viewed using the Cisco IOS command `show ip cef adjacency discard`, as shown in [Example 2-18](#).

Example 2-18. show ip cef adjacency discard Command

```
Router#show ip cef adjacency discard
Prefix           Next Hop           Interface
192.168.1.0/24   attached           Loopback0
```

Null Adjacency

CEF uses null adjacencies to indicate packets that are forwarded to the Null0 interface on a router. Destinations in the FIB that point to a null adjacency are dropped. This is commonly seen in the scenario where a static route pointing to Null0 is configured on an access server being accessed by numerous dialup users. As dialup users are connected to the access server host, routes are installed in the access server's routing table.

An example of a FIB entry pointing to a null adjacency is shown in [Example 2-19](#). ICMP unreachable messages are generated for null adjacencies.

Example 2-19. Null Adjacency

```
ip route 192.168.1.0 255.255.255.0 Null0
Router#show ip cef 192.168.1.0
192.168.1.0/24, version 37, epoch 0, attached
0 packets, 0 bytes
  via Null0, 0 dependencies
    valid null adjacency
```

No Route Adjacencies

If no default route is configured or learned through a routing protocol, CEF installs a valid no route adjacency for the 0.0.0.0/0 prefix, as shown in [Example 2-20](#). ICMP messages are generated for this type of adjacency.

Example 2-20. No Route Adjacency

```
Router#show ip route 0.0.0.0
% Network not in table

Router#show ip cef 0.0.0.0 0.0.0.0
0.0.0.0/0, version 0, epoch 0, attached, default route handler
0 packets, 0 bytes
  via 0.0.0.0, 0 dependencies
    valid no route adjacency
```

Cached and Uncached Adjacencies

A cached adjacency is a pointer reference to an adjacency used by the FIB entry to forward traffic. Destination prefixes with multiple next hops are not cached because multiple adjacencies can be used. [Example 2-21](#) illustrates cached and uncached adjacencies. The FIB entry for 192.168.1.1 shows two paths toward this destination and that the entry is not cached. On the other hand, the FIB entry for 192.168.1.2 indicates only one path to this destination and that the adjacency is cached.

Example 2-21. Cached and Uncached Adjacencies

Code View: [Scroll](#) / [Show All](#)

```
Router#show ip route
S          192.168.1.1 [1/0] via 172.16.17.4
                                     [1/0] via 172.16.17.5
S          192.168.1.2 [1/0] via 172.16.17.4

Router#show ip cef 192.168.1.1
192.168.1.1/32, version 48, per-destination sharing
0 packets, 0 bytes
  via 172.16.17.4, 0 dependencies, recursive
    traffic share 1
    next hop 172.16.17.4, Ethernet3/2 via 208.0.17.4/32
    valid adjacency
  via 172.16.17.5, 0 dependencies, recursive
    traffic share 1
    next hop 172.16.17.5, Ethernet3/2 via 208.0.17.5/32
    valid adjacency
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
```

```
Router#show ip cef 192.168.1.2
192.168.1.2/32, version 49, cached adjacency 172.16.17.4
0 packets, 0 bytes
  via 172.16.17.4, 0 dependencies, recursive
    next hop 172.16.17.4, Ethernet3/2 via 172.16.17.4/32
    valid cached adjacency
```

Unresolved Adjacency

An adjacency is unresolved if the adjacency for the next-hop IP address does not exist in the adjacency table. In [Example 2-22](#), the unresolved adjacency is the result of an incorrect configuration. The static route recursively points 192.168.2.0/24 to 192.168.1.0, which is the address of the connected network toward the destination. However, when configuring a recursive static route to a destination, the next-hop IP address configured should be the host IP address of the connected next-hop device, not the network address. In this scenario, packets destined for the 192.168.2.0/24 network will be dropped by CEF. To be more specific, if the result of recursive resolution points to a special adjacency, a receive in this case, an unresolved adjacency is employed.

Example 2-22. Unresolved Adjacency

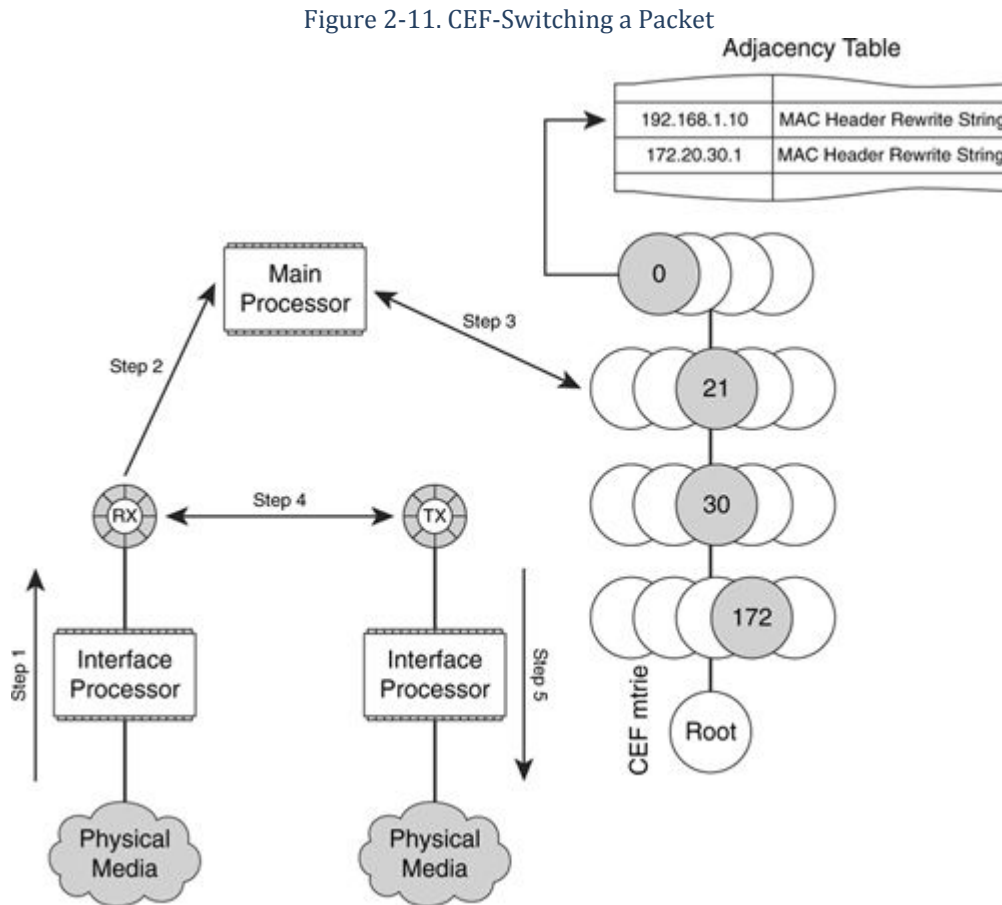
```
Router#show ip route 192.168.2.0
Routing entry for 192.168.2.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 192.168.1.0
    Route metric is 0, traffic share count is 1

Router#show ip route 192.168.1.0
Routing entry for 192.168.1.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
  * directly connected, via Ethernet0/0
    Route metric is 0, traffic share count is 1

Router#show ip cef 192.168.2.0
192.168.2.0/24, version 12, epoch 0
0 packets, 0 bytes
  via 192.168.1.0, 0 dependencies, recursive
  unresolved
```

Switching a Packet with CEF

Figure 2-11 illustrates the process of CEF-switching a packet.



The following list provides a step-by-step explanation of how a packet is switched in the CEF path:

1. The router's interface processor reads in the packet from the network media and stores it in buffer memory, called the interface's receive (RX) ring.
2. The router's interface processor sends a receive interrupt to the processor. The rest of the switching takes place during this interrupt.
3. The router performs a longest-match lookup for the destination in the FIB, using the destination IP address as the search key. If the FIB lookup is unsuccessful, the packet is dropped. If the FIB lookup is successful, a FIB path is chosen. The router follows the pointer from the chosen FIB path to the corresponding adjacency entry.
4. The router rewrites the Layer 2 header rewrite using the encapsulation string from the adjacency table and places the packet on the correct output queue for transmission on the corresponding outbound interface.
5. A packet that has been successfully switched is then enqueued on the outbound interface's transmit (TX) ring.

The CEF Epoch

The FIB and adjacency tables have a concept of a global version, referred to as the epoch. Entries added or modified have their epoch number set to the table's epoch number. Epoch numbers occupy a byte and range from 0-255, and their primary use is for high availability in large-scale networks. During a switchover, the CEF tables have their global epoch number incremented. The entries are then updated from converging routing protocols

and the entries get the new epoch number. After all the routing protocols have converged, a purge operation deletes any entries that do not have the current global epoch number. This process was introduced with the Nonstop Forwarding Enhanced FIB Refresh feature, which provides routers with the capability to continue forwarding IP traffic while the CEF tables are being rebuilt. IP forwarding on the router is therefore uninterrupted.

Configuring CEF/dCEF

To enable CEF switching on all interfaces, use the following command in global configuration mode:

```
Router(config)#ip cef
```

To enable or disable dCEF switching on all interfaces, use the following command in global configuration mode:

```
Router(config)#ip cef distributed
```

To disable CEF or dCEF switching for packets received on an interface, use the following command in interface configuration mode:

```
Router(config-if)#no ip route-cache cef
```

When CEF or dCEF has been disabled on an interface, Cisco IOS Software switches packets using the next fastest switching path. In the case of CEF, the next fastest switching path is fast switching. In the case of dCEF, the next fastest switching path is CEF on the route processor. On most high-end routing platforms, dCEF is required and cannot be disabled. CEF or dCEF should never be disabled unless specifically directed by a Cisco technical support representative.

To confirm that CEF or dCEF has been configured and is enabled, enter the `show ip cef` command, as shown in [Example 2-23](#).

Example 2-23. CEF Disabled Versus Enabled

Code View: [Scroll](#) / [Show All](#)

```
Router#show ip cef
%CEF not running
Router#show ip cef
Prefix                Next Hop                Interface
0.0.0.0/0             drop                    Null0 (default route handler
entry)
0.0.0.0/32            receive
172.28.6.72/30        attached                Serial2/0
172.28.6.72/32        receive
172.28.6.74/32        receive
172.28.6.75/32        receive
172.30.4.0/25         attached                Ethernet0/0
172.30.4.0/32         receive
172.30.4.2/32         receive
172.30.4.127/32       receive
224.0.0.0/4           drop
224.0.0.0/24          receive
255.255.255.255/32    receive
```

Chapter 3. CEF Enhanced Scalability

This chapter covers the following topics:

- [Fundamental changes to CEF for CSSR](#)
- [Changes to show commands](#)
- [New show ip cef commands](#)
- [New show cef commands](#)
- [CEF event logger](#)
- [CEF consistency checker](#)
- [New CEF processes](#)

Cisco Express Forwarding (CEF) is the most widely used forwarding mechanism on IP networks, so why change what works? There are several reasons:

- To improve CEF scaling and convergence times, with the end goal being the ability to handle up to 1 million prefixes in the forwarding table (so that you can sleep well if your network has reached the 900,000-route mark, and you were worried about the next phase of network growth)
- To make the interfaces between CEF, the routing table, access lists, Multiprotocol Label Switching (MPLS), and the various hardware forwarding engines more consistent and more defined
- To improve memory utilization
- To provide a more consistent mechanism to add new features to the switching path
- To provide for CEF Management Information Base (MIB) support
- To improve the performance of MPLS traffic engineering (TE) switching
- To merge the IP version 4 (IPv4) CEF tables and the IPv6 CEF tables, and their associated infrastructure and control interfaces

Modifications to CEF that occurred in Cisco IOS Release 12.2S largely involve internal changes that aren't obvious through output at the console, but improve the rate at which new features can be introduced, decrease the amount of work required to make CEF work with new hardware, and increase the quality of the code in the switching path.

This chapter starts by discussing fundamental changes to CEF to implement CEF Enhanced Scalability (also called CSSR), including new data structures that are slightly more complex than the ones we describe in [Chapter 2](#), "Cisco Express Forwarding." The output from show commands has changed somewhat in this newer version of CEF; that's the next item on the menu, followed by a very useful new feature, the CEF event logger, and finally, a high-level overview of some new CEF processes.

Fundamental Changes to CEF for CSSR

Two primary changes were made to fundamental CEF operation for CSSR:

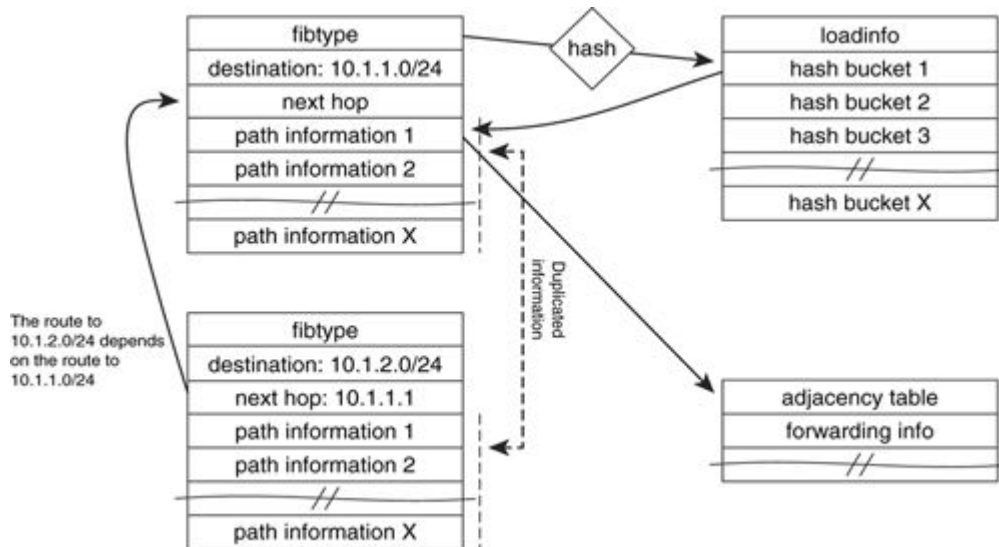
- The data structures making up the CEF tables
- The switching path packets take when switched by CEF

The following sections cover these two areas of fundamental change.

Data Structures

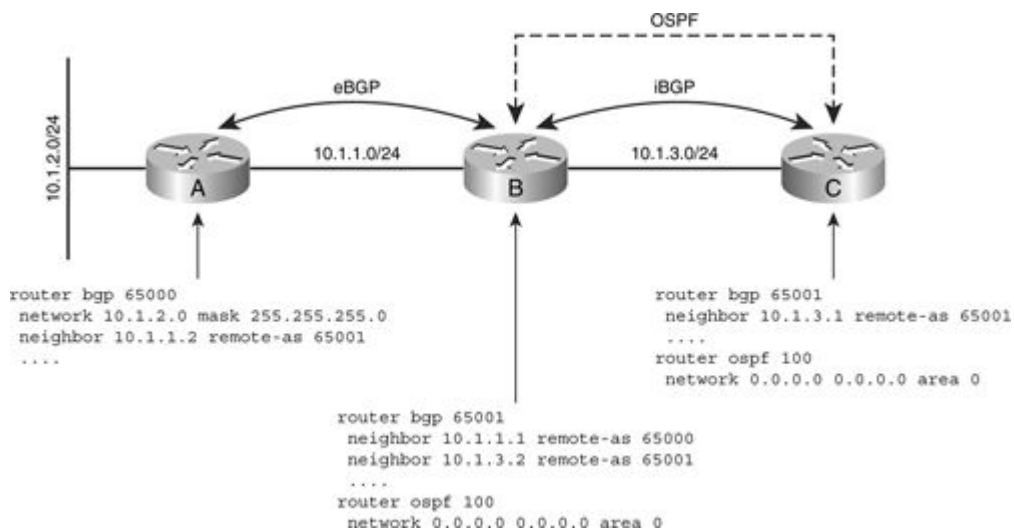
Before CSSR, CEF held data using three different types of data structures: the Forwarding Information Base (FIB), the loadinfo, and the adjacency. These data structures contained some amount of overlapping information, which was copied between them when some types of CEF entry were created or modified. To save memory when storing forwarding information, CSSR added a new, fourth type of data structure, which contains a single copy of the replicated information from the older data structures. Figure 3-1 illustrates CEF before CSSR.

Figure 3-1. CEF Data Structures
[View full size image]



In this case, the route to 10.1.2.0/24 uses 10.1.1.1 as its next hop. The 10.1.1.1 network is not, itself, directly connected but is reachable through some other next hop. This is called a routing recursion, a common occurrence in many networks. As Figure 3-1 shows, in this case, the path information for 10.1.1.0/24 and 10.1.2.0/24 are identical, because both routes are reachable through the same adjacency table entries. Figure 3-2 shows how path recursion arises in a network.

Figure 3-2. Route Recursion in a Network
[View full size image]



In [Figure 3-2](#), Routers B and C are running Open Shortest Path First (OSPF) on all their interfaces. This means that Router C has a route, learned through OSPF, for the 10.1.1.0/24 network, which connects Routers A and B.

Router A is also configured to advertise 10.1.2.0/24 to Router B through Border Gateway Protocol (BGP) across an external BGP (eBGP) peering session. Router B is readvertising 10.1.2.0/24 through BGP, across an internal BGP (iBGP) peering session, to Router C. When Router B is readvertising 10.1.2.0/24 through BGP to Router C, it leaves the next hop toward the destination set to the IP address of the BGP it learned the route from—in this case, Router A—or an address on the 10.1.1.0/24 network.

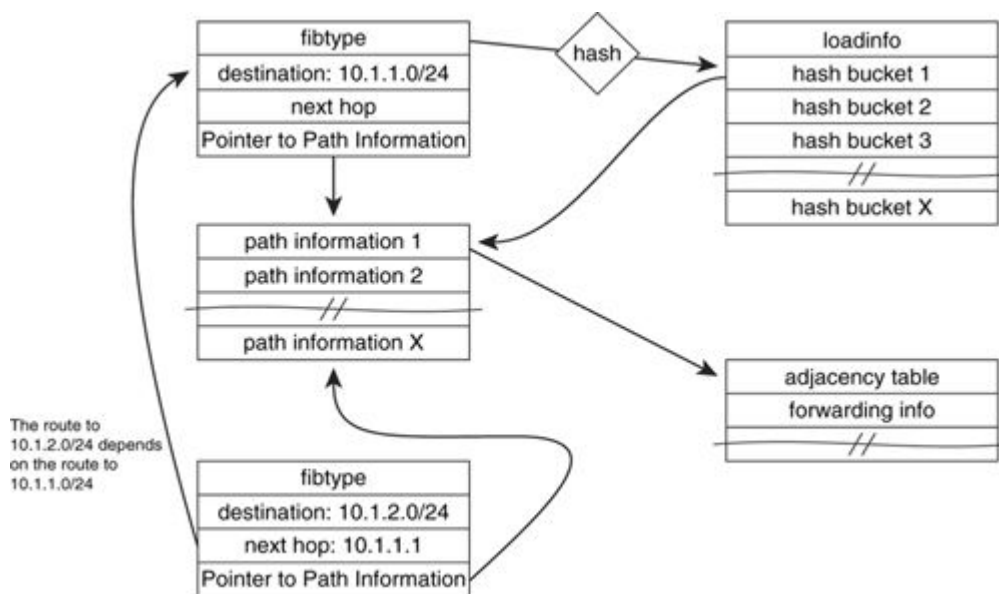
For Router C to forward traffic to a destination on 10.1.2.0/24, it looks up the destination and finds it is reachable through a next hop on the 10.1.1.0/24 network. This is the route recursion on the network that [Figure 3-1](#) shows in the CEF table. Router C uses the path to 10.1.1.0/24 to reach destinations on 10.1.2.0/24.

Because BGP normally does not reset the next hop on routes received from eBGP peers, route recursion is normal in large-scale networks using BGP on top of some other routing protocol, such as most service provider networks.

To reduce the amount of memory CEF uses in networks of this type, CSSR separates the path information out from the remainder of the FIB information, including the destination and prefix information. This allows recursive routes to share the same path information, reducing memory requirements, as shown in [Figure 3-3](#).

Figure 3-3. CEF Data Structures After CSSR

[\[View full size image\]](#)



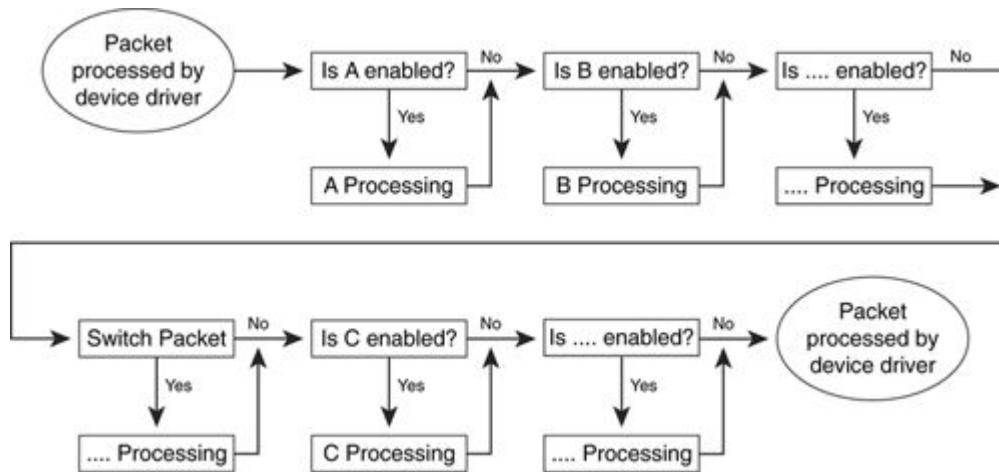
As you can see from [Figure 3-3](#), only one copy of the path information is stored in this new structure design. This reduces duplicated information in the CEF table, especially in networks with the largest number of routes, which use BGP and hence have a lot of routes reachable through recursion.

Switching Path Changes

CSSR also made one change to the CEF switching path. In pre-CSSR CEF, the switching path was a monolithic unit; each feature was checked as a packet was switched through the router in software, as [Figure 3-4](#) illustrates.

Figure 3-4. CEF Switching Path

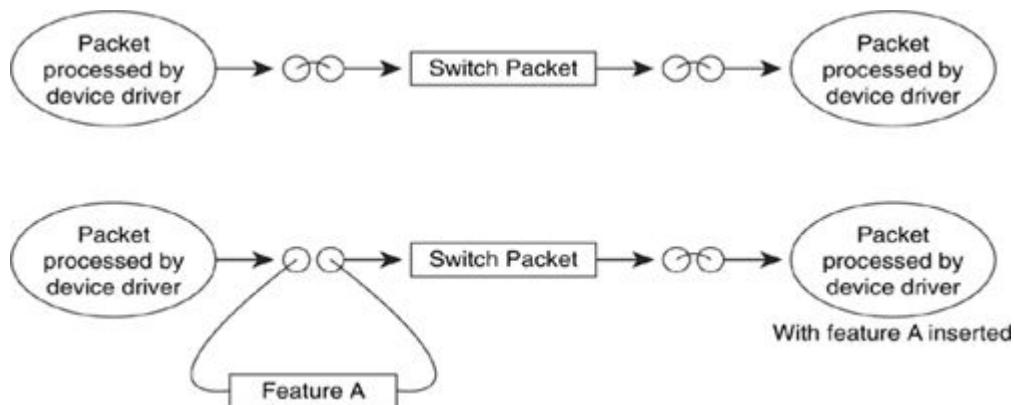
[\[View full size image\]](#)



In pre-CSSR CEF, each feature in the CEF switching path, including Network Address Translation (NAT), packet filtering, and Web Cache Communications Protocol (WCCP), is checked as the packet is switched, regardless of whether the feature is enabled. If the feature is not enabled, the check is simple—just one or two lines of code—but for each feature added to the CEF switching path, the CEF code itself must be changed and maintained.

CSSR changed this process, creating an insertion point in the switching path, as illustrated in Figure 3-5.

Figure 3-5. Feature Insertion in CSSR
[\[View full size image\]](#)



Normally, packets are switched along the path from the inbound device driver, through the CEF switching code, to the outbound device driver. No checks are made to determine whether additional features require processing on the packet.

If a feature is required, a new node is inserted in the switching path. As the packet is switched, control is passed to this additional node in the switching chain and then back to the CEF process. Multiple nodes can be inserted either before or after the CEF switch itself; they are chained, so each one occurs before or after some other feature in the switching path.

This change in the forwarding path not only eliminates the requirement to check for each feature in the path being enabled, but it also allows new features to be added, or old features to be removed, without modifying the CEF code responsible for switching packets.

Changes to show Commands

A number of commands have been added or changed in CSSR. The following sections describe each change or added show command.

show ip cef

Example 3-1 shows the output of the `show ip cef` command in CEF and CSSR. The output is much shorter than in pre-CSSR CEF, because some information has been taken out of the command. The version, epoch, packet count, and byte count have all been removed, because this information is available elsewhere. The wording of the output has been shortened as well.

Example 3-1. Output of the `show ip cef` Command

```
CEF-router#show ip cef 10.1.1.0 detail
10.1.1.0/24, epoch 0
  via 10.1.2.1, 0 dependencies, recursive
    next hop 10.1.2.1, Ethernet0/0 via 10.1.2.0/24
    valid glean adjacency

CSSR-router#show ip cef 10.1.1.0 detail
10.1.1.0/24, epoch 0
  recursive via 10.1.2.1
  attached to FastEthernet0/0
```

show ip cef interface

Most of the information in the header of the `show ip cef [interface] detail` command has been removed, because it is available elsewhere; this includes the number of routes, number of leaves, the type of load sharing configured, the number of resets, and the timer configuration. The display of the CEF table entries in the second part of the output is similar to the output format for `show ip cef`, with the same changes as noted previously. You can see these changes in **Example 3-2**.

Example 3-2. Output of the `show ip cef detail` Command

Code View: [Scroll](#) / [Show All](#)

```
CEF-router#show ip cef [interface] detail
IP CEF with switching (Table Version 66), flags=0x0
 46 routes, 0 reresolve, 0 unresolved (0 old, 0 new), peak 1
 46 leaves, 23 nodes, 30360 bytes, 85 inserts, 39 invalidations
 0 load sharing elements, 0 bytes, 0 references
 universal per-destination load sharing algorithm, id 51DB3C24
 3(0) CEF resets, 0 revisions of existing leaves
 Resolution Timer: Exponential (currently 1s, peak 1s)
 0 in-place/0 aborted modifications
 refcounts: 3167 leaf, 3112 node

Table epoch: 0 (46 entries at this epoch)

Adjacency Table has 5 adjacencies
10.1.2.0/24, version 43, epoch 0, attached, connected
0 packets, 0 bytes
  via Ethernet0/0, 0 dependencies
  valid glean adjacency
....

CSSR-router#show ip cef [interface] detail
```

```
IPv4 CEF is enabled and running
VRF Default:
 31 prefixes (31/0 fwd/non-fwd)
 Table id 0, version 31, 1 resets
 Database epoch: 0 (31 entries at this epoch)

10.1.2.0/24, epoch 0, flags attached, connected
 attached to FastEthernet0/0
10.1.2.1/32, epoch 0, flags adjfib
 NetFlow: Origin AS 0, Mask Bits 0
 attached to FastEthernet0/0
10.1.1.0/24, epoch 0
 recursive via 10.1.2.1
 attached to FastEthernet0/0
```

show ip cef summary

A number of fields have been removed from the show ip cef summary output as well, because the information is available in the output of other show commands. The output of show ip cef summary is identical to the header section of show ip cef [interface] detail, described in the preceding section, and the changes are identical. [Example 3-3](#) provides a sample output for show ip cef summary.

Example 3-3. Output of the show ip cef summary Command

```
CEF-router#show ip cef summary
IP CEF with switching (Table Version 66), flags=0x0
 46 routes, 0 reresolve, 0 unresolved (0 old, 0 new), peak 1
 46 leaves, 23 nodes, 30360 bytes, 85 inserts, 39 invalidations
 0 load sharing elements, 0 bytes, 0 references
 universal per-destination load sharing algorithm, id 51DB3C24
 3(0) CEF resets, 0 revisions of existing leaves
 Resolution Timer: Exponential (currently 1s, peak 1s)
 0 in-place/0 aborted modifications
 refcunts: 3167 leaf, 3112 node

 Table epoch: 0 (46 entries at this epoch)

CSSR-router#show ip cef summary
IPv4 CEF is enabled and running
VRF Default:
 31 prefixes (31/0 fwd/non-fwd)
 Table id 0, version 31, 1 resets
 Database epoch: 0 (31 entries at this epoch)
```

show cef state capabilities

The output of the show cef state command has changed dramatically in CSSR, as shown in [Example 3-4](#). The new output shows three sections: one for the route processor (RP instance), one for IPv4 (IPv4 CEF Status), and one for IPv6 (IPv6 CEF Status). Much of the information included in the older version of the output has been removed, because it is available in the output of other show commands.

Example 3-4. Output of the show ip cef state capabilities Command

Code View: [Scroll](#) / [Show All](#)

```
CEF-router#show cef state capabilities
CEF Status [RP]
  CEF enabled/running
  dCEF disabled/not running
  CEF switching enabled/running
  CEF default capabilities:
    Always CEF switching:          no
    Always dCEF switching:         no
    Default CEF switching:         yes
    Default dCEF switching:        no
    Drop multicast packets:        no
    OK to punt packets:            yes
    NVGEN CEF state:               no
    fastsend() used:               yes
    CEF NSF capable:               no
    RPR+/SSO standby capable:      no
    IPC delayed func on SSO:       no
    FIB auto repair supported:     yes
    LCs not running at init time:  no
    Hardware forwarding supported:  no
    Hardware forwarding in use:    no
    Load-sharing pr. packet supported: yes

CSSR-router#show cef state
CEF Status:
  RP instance
  common CEF enabled
IPv4 CEF Status:
  CEF enabled/running
  dCEF disabled/not running
  CEF switching enabled/running
  universal per-destination load sharing algorithm, id 5E850505
IPv6 CEF Status:
  CEF disabled/not running
  dCEF disabled/not running
  original per-destination load sharing algorithm
```

New show ip cef Commands

Three new commands have been added under show ip cef, including show ip cef tree, show ip cef internal, and show ip cef switching statistics. These three commands are covered in the following sections.

show ip cef tree

Example 3-5 shows the output of the show ip cef tree command.

Example 3-5. Output of the show ip cef tree Command

```
CSSR-router#show ip cef tree
VRF Default tree information:
MTRIE/RTREE storing IPv4 addresses
31 entries (31/0 fwd/non-fwd)
Forwarding tree:
  Forwarding lookup routine: IPv4 mtrie 8-8-8-8 optimized
  36 inserts, 5 deletes
  8-8-8-8 stride pattern
  short mask protection enabled for <= 4 bits without process suspension
  31 leaves (868 bytes), 20 nodes (20800 bytes)
  23236 total bytes
  leaf ops: 36 inserts, 5 deletes
  leaf ops with short mask protection: 3 inserts, 2 deletes
  per-prefix length stats: lookup off, insert off, delete off
  refcounts: 2150 leaf, 2085 node
  node pools:
    pool[C/8 bits]: 20 allocated (0 failed), 20800 bytes
Non-Forwarding tree:
  29 inserts, 29 deletes
  0 nodes using 0 bytes
```

The output of show ip cef tree includes the following:

- A section per the virtual routing and forwarding (VRF) table. In the output in this example, there is only one VRF, the Default, so only one section is shown. If more than one VRF existed on this router, there would be a section for each VRF configured.
- The type of addresses, IPv4 or IPv6, being stored in the tree.
- The number of entries in the tree.
- The type of tree used in this VRF, which indicates the stride. In [Example 3-5](#), the stride is 8-8-8-8.
- The number of nodes and leaves in the tree, as well as the number of tree operations, such as inserts and deletes.
- Information about the amount of memory used by each tree.

show ip cef internal

Another new command added is show ip cef internal, as shown in [Example 3-6](#). If you run the show ip cef internal command, you might see some entries prefixed by the tilde symbol (~). These are nonforwarding entries, which are stored in a separate tree from the forwarding entries. Nonforwarding entries result from CEF receiving information about specific destinations from multiple sources and only using one of the provided forwarding entries.

Example 3-6. Output of the show ip cef internal Command

Code View: [Scroll](#) / [Show All](#)

```
CSSR-router #show ip cef internal
IPv4 CEF is enabled and running
VRF Default:
 31 prefixes (31/0 fwd/non-fwd)
 Table id 0, version 31, 1 resets
 Database epoch: 0 (31 entries at this epoch)

0.0.0.0/32, version 0, epoch 0, flags receive, RIB, refcount 4
 sources: CEF
 path 638F83F0, path list 638F3350, share 1, flags receive
 ifnums: (none)
 path_list contains no resolved destination(s). HW IPv4 notified.
 receive
 output chain: receive (10)
10.1.2.0/24, version 4, epoch 0, flags attached, connected, RIB, refcount 4
 sources: RIB
 feature space:
 IPRM: 0x0004800C
 path 638F82A0, path list 638F3230, share 1, flags attached
 ifnums: (none)
 path_list contains at least one resolved destination(s). HW IPv4
notified.
 attached to FastEthernet0/0, adjacency glean
 output chain: glean
```

The `show ip cef internal` command essentially shows each CEF table entry, with all the available information about the entry. Useful fields include the source of the CEF table entry or the process that installed the entry, the type of entry, interfaces, and features installed on the switching path for each entry. This is a lot of information, and it probably won't be useful in many troubleshooting situations.

show ip cef switching statistics

CSSR adds a new command, `show [ip|ipv6] switching statistics [feature]`. This new show command provides information on each packet that CEF switched. For any packets punted to the process-switching path, the output of `show ip cef switching` will tell you why they were punted. [Example 3-7](#) provides sample output for this new command.

Example 3-7. Output of the show ip cef switching statistics Command

```
CSSR-Router# show ip cef switching statistics
```

Path	Reason	Drop	Punt	Punt2Host
RP RIB	Packet destined for us	0	253	0
RP RIB	Total	0	253	0
RP LES	Packet destined for us	0	253	0
RP LES	Total	0	253	0
RP PAS	Packet destined for us	0	506	0
RP PAS	TTL expired	0	0	160
RP PAS	Total	0	506	160
All	Total	0	1012	160

New show cef Commands

Another series of commands under show cef have also been added as part of CSSR. These commands contain generic information about CEF, rather than information related to IP switching of CEF.

As shown in [Example 3-8](#), the output of the show cef fib and show cef loadinfo commands provides information about the number of entries allocated in each table and memory failures encountered when allocating new entries. This information is useful when troubleshooting a router with low memory or a memory leak.

Example 3-8. Output of the show cef fib and show cef loadinfo Commands

```
CSSR-router#show cef fib
31 allocated IPv4 entries, 0 failed allocations
0 allocated IPv6 entries, 0 failed allocations

CSSR-router#show cef loadinfo
0 allocated loadinfos, 0 failed allocations
```

In [Example 3-9](#), the output of show cef memory shows each type of memory CEF uses and how much of it is in use. This is useful information for troubleshooting memory leaks or a router with a memory allocation problem.

Example 3-9. Output of the show cef memory Command

```
CSSR-router#show cef memory
Memory                               in use/allocated          Count
-----
ADJ: DROP adjacency                  :      368/424             ( 86%) [1]
ADJ: Discard adjacency                :      368/424             ( 86%) [1]
....
CEF: FIBHWIDB                         :      7592/8320           ( 91%) [13]
CEF: FIBIDB                          :       2600/3328           ( 78%) [13]
CEF: FIBSWB control                   :       576/1024           ( 56%) [8]
....
```

The new command show cef table provides a summary of each CEF table configured on the router. [Example 3-10](#) shows two CEF tables configured: a single table for IPv4 forwarding information and a single table for IPv6 forwarding information. The number of prefixes and the table version number are given. Any VRF within each table is listed in the table.

Example 3-10. Output of the show cef table Command

```
CSSR-router#show cef table
1 active IPv4 table out of a maximum of 10000
VRF          Prefixes      Version      Memory  Flags
Default          31          31          25584

1 active IPv6 table out of a maximum of 1
VRF          Prefixes      Version      Memory  Flags
Default          0           0           72
```

[Example 3-11](#) shows the output of show cef timers, which provides a somewhat graphical display of the timers used to maintain the CEF tables. Timers in Cisco IOS Software are related by a parent/child relationship; when the parent timer expires (wakes up), all the child timers are marked as expired as well. This allows multiple

overlapping events to be controlled independently but resynchronized, or easily restarted at the same time, when certain events occur.

Example 3-11. Output of the show cef timers Command

```
CSSR-router#show cef timers
CEF background process
  Expiration    Type
|      18.196  (parent)
|      18.196  FIB checkers: IPv4 scan-rib-ios scanner
|      18.196  FIB checkers: IPv4 scan-ios-rib scanner
|      18.196  FIB checkers: IPv6 scan-ios-rib scanner
|      18.468  FIB checkers: IPv4 scan-hw-sw scanner
|      18.468  FIB checkers: IPv4 scan-sw-hw scanner

Platform counter polling is not enabled
IPv4 CEF background process
  Expiration    Type
|      0.160  (parent)
|      0.160  adjacency update hwidb
|      0.196  ARP throttle
|      3.192  fibidb queue
```

Most of the timers relate to checking the CEF tables for consistency periodically or with throttling certain types of responses, such as Address Resolution Protocol (ARP) requests, so that they don't occur too often.

Finally, an entire chain of new commands allow you to examine the path information that has been added, including show cef path, show cef path list, and show cef path list walk. Each one shows the same information, with increasing amounts of detail, as [Example 3-12](#) shows.

Example 3-12. Output of the show cef path Command

```
Code View: Scroll / Show All

CSSR-router#show cef path
39 allocated IPv4 paths, 0 failed allocations
0 allocated IPv6 paths, 0 failed allocations

39 Total Paths, 1 Recursive Paths, 0 Unresolved Paths

CSSR-router#show cef path list
38 path lists (11 in shared path list hash table, 27 in special list)
0 failed allocations

hash table:
[ 2] path list 638F2870, 1 path, 1 output chain, 1 lock
[ 4] path list 638F3230, 1 path, 1 output chain, 1 lock
[ 8] path list 638F30B0, 1 path, 1 output chain, 1 lock
[ 9] path list 638F2F30, 1 path, 1 output chain, 1 lock
[10] path list 638F2DB0, 1 path, 1 output chain, 1 lock
[11] path list 638F2C30, 1 path, 1 output chain, 1 lock
[12] path list 638F28D0, 1 path, 1 output chain, 1 lock
[12] path list 638F2930, 2 paths, 1 output chain, 1 lock
[12] path list 638F2AB0, 1 path, 1 output chain, 1 lock
[31] path list 638F2510, 1 path, 1 output chain, 3 locks
[46] path list 638F2690, 1 path, 1 output chain, 3 locks

CSSR-router#show cef path list walk
CSSR-router##show cef path list walk
```

```
38 path lists (11 in shared path list hash table, 27 in special list)
0 failed allocations
```

```
hash table:
```

```
[ 2] path list 638F2870, 1 path, 1 output chain, 1 lock
[ 4] path list 638F3230, 1 path, 1 output chain, 1 lock
[ 8] path list 638F30B0, 1 path, 1 output chain, 1 lock
[ 9] path list 638F2F30, 1 path, 1 output chain, 1 lock
[10] path list 638F2DB0, 1 path, 1 output chain, 1 lock
[11] path list 638F2C30, 1 path, 1 output chain, 1 lock
[12] path list 638F28D0, 1 path, 1 output chain, 1 lock
[12] path list 638F2930, 2 paths, 1 output chain, 1 lock
[12] path list 638F2AB0, 1 path, 1 output chain, 1 lock
[31] path list 638F2510, 1 path, 1 output chain, 3 locks
[46] path list 638F2690, 1 path, 1 output chain, 3 locks
```

```
hash table path lists:
```

```
path list 638F2870, flags 21, 2 locks
ifnums: (none)
  path_list contains no resolved destination(s). HW IPv4 notified.
1 path
  path 638F76D0, path list 638F2870, share 1, flags attached
  ifnums: (none)
  path_list contains no resolved destination(s). HW IPv4 notified.
  attached to Null0, adjacency Null0
1 output chain
chain[0]: Null0
path list 638F3230, flags 29, 2 locks
ifnums: (none)
  path_list contains at least one resolved destination(s). HW IPv4
notified.
....
```

CEF Event Logger

Cisco IOS Software components include event loggers. An event logger is a process that runs constantly, collecting much of the same information that various types of debug output provide, but without the overhead and without having to be explicitly enabled.

The event logger allows you to gather the information required to troubleshoot a problem regardless of whether you explicitly enabled debugging when the problem occurred, as long as you catch the log soon after the problem occurs. Event logs generally have a fixed size, which means that they will only hold a specific number of events before discarding the oldest event to replace it with the most recent one. The size of the IP CEF event log is set using the `ip cef table event-log` command; the default size is 16,000 events.

The CEF event logger records events in the CEF table, such as the insertion and deletion of CEF entries, as shown in [Example 3-13](#).

Example 3-13. Output of the show ip cef event Command

Code View: [Scroll](#) / [Show All](#)

```
CSSR-router#show ip cef event
% Command accepted but obsolete, unreleased or unsupported; see
documentation.
```

```

00:00:09.380: [Default] *.*.*./*'00          New FIB table          [OK]
00:00:11.112: [Default] 0.0.0.0/32'00          FIB insert             [OK]
00:00:11.112: [Default] 255.255.255.255/32'00 FIB insert             [OK]
00:00:11.112: [Default] 224.0.0.0/24'00          FIB insert             [OK]
00:00:11.112: [Default] 224.0.0.0/4'00           FIB insert             [OK]
00:00:11.332: [Default] 224.0.0.0/4'00          FIB remove (flagged)  [OK]
00:00:11.332: [Default] 224.0.0.0/4'00          FIB remove (deleted)  [OK]
00:00:11.332: [Default] 224.0.0.0/4'00          FIB insert             [OK]
00:00:11.584: [Default] 0.0.0.0/32'00          FIB remove (flagged)  [OK]
00:00:11.584: [Default] 0.0.0.0/32'00          FIB remove (deleted)  [OK]
00:00:11.584: [Default] 224.0.0.0/24'00          FIB remove (flagged)  [OK]
00:00:11.584: [Default] 224.0.0.0/24'00          FIB remove (deleted)  [OK]
00:00:11.584: [Default] 224.0.0.0/4'00           FIB remove (flagged)  [OK]
00:00:11.584: [Default] 224.0.0.0/4'00           FIB remove (deleted)  [OK]
00:00:11.584: [Default] 255.255.255.255/32'00 FIB remove (flagged)  [OK]
00:00:11.584: [Default] 255.255.255.255/32'00 FIB remove (deleted)  [OK]
00:00:11.584: [Default] *.*.*./*'00          Flush FIB table (4/0ms)
[OK]
00:00:11.584: [Default] 0.0.0.0/32'00          FIB insert             [OK]
00:00:11.584: [Default] 255.255.255.255/32'00 FIB insert             [OK]
00:00:11.584: [Default] 224.0.0.0/24'00          FIB insert             [OK]
00:00:11.584: [Default] 224.0.0.0/4'00           FIB insert             [OK]
00:00:11.588: [Default] 10.1.2.0/24'00           FIB insert             [OK]
00:00:11.588: [Default] 10.1.2.21/32'00          FIB insert             [OK]
00:00:11.588: [Default] 10.1.2.0/32'00          FIB insert             [OK]
00:00:11.588: [Default] 10.1.2.255/32'00         FIB insert             [OK]
00:00:11.588: [Default] 10.1.2.0/24            NBD up                 [OK]
....

```

The first line indicates that this is an unsupported command, which means you will not find much documentation about this command on Cisco.com or in any manuals. This is because this command will eventually be obsolete, replaced by commands under the monitor event-log chain, as shown in the following example.

The columns in this output contain the following information:

- **Timestamp**— This is the time at which the event occurred.
- **VRF**— The name of the VRF in which the event occurred is contained in brackets.
- **Prefix and Prefix Length**— The prefix that was inserted or removed, or that some other action was taken on is contained in this column.
- **Action**— The action taken is contained in this column.
- **Result**— The result of the action is contained in this column.

The action column can contain a large number of values, including the following:

- Events concerning the enabling or running of the CEF process, such as FIB enabled, FIB running, distributed FIB (dFIB) enabled, and dFIB running
- Error-handling events, such as "Handling malloc failed"
- CEF process events, such as "Scanner process created" and "Scanner event loop enter"
- IPv4 FIB table entry events, such as FIB insert and FIB delete

- IPv6 FIB table entry events, such as FIB insert and FIB delete
- Line card events driven by the route processor
- Adjacency table events, such as interface up or down, protocol up or down, and others

Another CEF event trace log is also accessible through the `show monitor event-trace cef` command, as shown in [Example 3-14](#).

Example 3-14. Output of the `show monitor event-trace cef` Command

Code View: [Scroll](#) / [Show All](#)

```
CSSR-router#show monitor event-trace cef ?
  all          Show all the traces in current buffer
  back         Show trace from this far back in the past
  clock        Show trace from a specific clock time/date
  events       CEF Events
  from-boot    Show trace from this many seconds after booting
  interface    CEF Interface Events
  ipv4         CEF IPv4 Events
  ipv6         CEF IPv6 Events
  latest       Show latest trace events since last display
  merged       Show entries in all event traces sorted by time

CSSR-router##show monitor event-trace cef all

cef_events:

00:00:03.172: Inst      unknown -> RP
00:00:03.172: SubSys   fib_ios_chain init
00:00:09.372: SubSys   fib init
00:00:09.376: SubSys   ipv4fib init
00:00:09.384: SubSys   ipv4fib_ios init
00:00:09.432: SubSys   fib_ios init
....
cef_interface:

00:00:09.440: <empty>      (sw  3) Create      new
00:00:09.440: <empty>      (sw  3) SWIDBLnk FastEthernet0/0(3)
00:00:09.440: Fa0/0        (sw  3) NameSet
00:00:09.440: <empty>      (hw  1) Create      new
00:00:09.440: <empty>      (hw  1) HWIDBLnk FastEthernet0/0(1)
00:00:09.440: Fa0/0        (hw  1) NameSet
00:00:09.440: Fa0/0        (sw  3) State        down -> up
00:00:09.440: <empty>      (sw  4) Create      new
00:00:09.440: <empty>      (sw  4) SWIDBLnk FastEthernet1/0(4)
....
```

As you can see from the output shown in [Example 3-13](#), this information is more useful for code-level debugging than CEF-level debugging.

CEF Consistency Checker

The CEF consistency checker verifies that the Update Manager (discussed in the section "New CEF Processes," later in this chapter) is maintaining the local FIB tables on each line card correctly. Two consistency checkers are included with CSSR: active and passive.

Passive Checkers

Passive checkers run constantly, in the background, unless you disable them using the [no] cef table consistency-check <ipv6|ipv4> command. During passive checking, the following items occur each minute:

- Each line card sends one interprocess communications (IPC) message containing CEF consistency checking information by default, although more can be configured.
- The route processor sends one IPC message containing CEF consistency check information to each line card.
- The route processor compares 1000 prefixes in the Routing Information Base (RIB) with their CEF entries to make certain that the CEF table matches the RIB. This is 60,000 prefixes per hour.

The configuration command `cef table consistency-check <af> type <scan -ios -rib> [count <count>] [period <seconds>]` controls the number of prefixes examined in each passive check and the time between passive checks.

To control the recording of error messages when an inconsistency is found, use the configuration command `cef table consistency-check <af> error-message`. In both of these commands, <af> is the address family, such as IPv4 or IPv6, you would like to configure.

Active Checkers

An active consistency check is initiated at the console, using the `test cef enable` command, followed by `test cef table consistency [detail]`, as shown in [Example 3-15](#).

Example 3-15. Output of the test cef enable and test cef table consistency Commands

```
CSSR-router#test cef enable
The use of TEST CEF commands will severely impact network performance
and stability and should be used with extreme caution. For safety,
execute the "test cef disable" command to disable this capability when
it is no longer required.

CSSR-router#test cef table consistency detail
full-scan-rib-ios: Checking IPv4 RIB to FIB consistency
full-scan-rib-ios: FIB checked 8 prefixes, and found 0 missing.
full-scan-ios-rib: Checking IPv4 FIB to RIB consistency
full-scan-ios-rib: Checked 8 FIB prefixes in 1 pass, and found 0 extra.
Error: Failed to run IPv6 full-scan-rib-ios checker
Error: Failed to run IPv6 full-scan-ios-rib checker
No IPv4 inconsistencies found, check took 00:00:00.004
No IPv6 inconsistencies found, check took 00:00:00.000
```

An active check on a table of 150,000 prefixes can take between 5 and 60 seconds.

Consistency-Checking Process

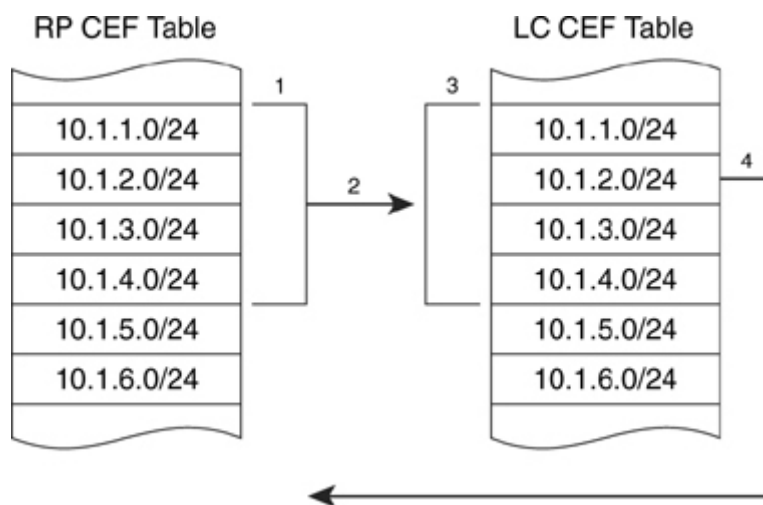
The consistency-checking process contains two phases:

- Building, transmitting, and comparing the FIB table information
- Handling a detected inconsistency

The following list and [Figure 3-6](#) illustrate the first part of this process:

1. The CEF consistency checker on the route processor builds a consistency check message by walking the local CEF table. For each entry, a description of the entry, including a checksum, is inserted into an IPC message.
2. When the IPC message is full, it is transmitted to all the line cards.
3. The CEF consistency checker compares the information received in the consistency check IPC packet with the same entries in the local CEF tables, including comparing the checksum computed locally with the checksum computed on the route processors.
4. If any entry does not match, the line card creates a new IPC message containing the local information about this entry and transmits it to the route processor.

Figure 3-6. CEF Consistency-Checking Process



When the route processor receives the IPC message from the line card, it reexamines the local CEF tables, looking for a mismatch. If the data is still mismatched, the route processor will build a new IPC packet with the correct information and transmit it to the line card. If, after three checks (which allows time for any pending updates to be completed), the line card and route processor tables continue to be inconsistent, the line card is marked inconsistent on the route processor.

After a line card is marked inconsistent, CEF can auto-repair the CEF tables, if `cef table consistency-check <af> auto-repair` is configured. To auto-repair, CEF waits 10 seconds to allow all current consistency checks to finish. At the end of this time, the CEF epoch is incremented. This, in turn, causes the route processor to walk through the local CEF tables, generating updates to every line card for every entry. As these updates are generated, old FIB table information is purged. A hold-down timer prevents multiple auto-repairs from running concurrently.

To check the current state of the CEF consistency checkers, use the `show cef table consistency-check` command, as shown in [Example 3-16](#).

Example 3-16. Output of the `show cef table consistency-check` Command

Code View: [Scroll](#) / [Show All](#)

```
CSSR-router#show cef table consistency-check
Consistency checker master control: enabled

IPv4:
Table consistency checker state:
  scan-rib-ios: disabled
  0/0/0/0 queries sent/ignored/checked/iterated
```

```
scan-ios-rib: disabled
 0/0/0/0 queries sent/ignored/checked/iterated
full-scan-rib-ios: enabled [1000 prefixes checked every 60s]
 0/0/0/0 queries sent/ignored/checked/iterated
full-scan-ios-rib: enabled [1000 prefixes checked every 60s]
 0/0/0/0 queries sent/ignored/checked/iterated
Checksum data checking disabled
Inconsistency error messages are disabled
Inconsistency auto-repair is enabled (10s delay, 300s holddown)
Inconsistency auto-repair runs: 0
Inconsistency statistics: 0 confirmed, 0/16 recorded
```

IPv6:

Table consistency checker state:

```
scan-ios-rib: disabled
 0/0/0/0 queries sent/ignored/checked/iterated
full-scan-rib-ios: enabled [1000 prefixes checked every 60s]
 0/0/0/0 queries sent/ignored/checked/iterated
full-scan-ios-rib: enabled [1000 prefixes checked every 60s]
 0/0/0/0 queries sent/ignored/checked/iterated
Checksum data checking disabled
Inconsistency error messages are disabled
Inconsistency auto-repair is enabled (10s delay, 300s holddown)
Inconsistency auto-repair runs: 0
Inconsistency statistics: 0 confirmed, 0/16 recorded
```

New CEF Processes

Beyond the data structure and switching path changes, which impact CEF switching directly, CSSR also added the following new processes to CEF to better manage the CEF data structures and CEF operation:

- [FIB Manager](#)
- [Adjacency Manager](#)
- [Update Manager](#)

The following sections describe each manager.

FIB Manager

The FIB Manager manages the prefix path lists and loadinfo maps, including managing multilevel load sharing (through recursions). The FIB Manager also dynamically allocates new memory chunks as needed and handles statistics and feature data.

Adjacency Manager

The Adjacency Manager, as its name implies, manages the adjacency tables. This includes managing the interface state, enabling and disabling protocols, and maintaining of a per-interface tree.

Update Manager

The Update Manager keeps track of which entries in the FIB tables need to be updated. Updates are pulled from the line cards to the route processor through the Update Manager, which allows the line cards to regulate the rate at which new FIB information is transferred.

Chapter 4. Basic IP Connectivity and CEF Troubleshooting

This chapter covers the following topics:

Troubleshooting IP connectivity

Troubleshooting punt adjacencies

Understanding CEF error messages

Troubleshooting commands reference

Cisco Express Forwarding (CEF) troubleshooting can be tedious, laborious, and difficult. However, most instances of CEF troubleshooting do not require detailed Cisco IOS architecture and platform (hardware) architecture knowledge. For example, many CEF issues are found in two or three steps of troubleshooting. In addition, many issues that appear to be CEF-related end up being a result of a misconfiguration or inoperable end device.

The first section of this chapter presents the general troubleshooting used on Cisco IOS routers and switches as a first step in troubleshooting IP connectivity problems. CEF occasionally is the scapegoat for IP connectivity problems, and this chapter helps you verify whether CEF is the root cause of a particular IP connectivity problem.

This chapter does not delve into platform specifics of troubleshooting CEF. The chapter simply approaches troubleshooting from a CEF software-switching and command-line interface (CLI) perspective. Most mid- to high-end routers and all Catalyst switches support distributed CEF (dCEF), or hardware switching.

This chapter begins the CEF troubleshooting for all Cisco platforms, including the Cisco 2600, 3700, 7500, 12000, and Catalyst 6500. Chapter 5, "Understanding Packet Switching on the Cisco Catalyst 6500 Supervisor 720," goes into further detail for additional platform and hardware troubleshooting of CEF for the Cisco Catalyst 6500.

The chapter concludes with a table of the basic CEF troubleshooting commands.

Troubleshooting IP Connectivity

As mentioned in the introduction, CEF is a common scapegoat for IP connectivity issues. As such, when approaching an IP connectivity issue, keep an open mind about the root cause of the issue.

This section reviews the methodology for troubleshooting IP connectivity issues, which leads to identifying and troubleshooting CEF issues.

Note

This chapter is based on Cisco IOS Release 12.3. All the command output presented in this chapter might not be available in previous releases of Cisco IOS. Check the command reference for your specific Cisco IOS version to verify whether a specific command is supported.

The best approach in troubleshooting is to build a troubleshooting plan. Flow charts simplify troubleshooting because they present a stepwise approach to troubleshooting. The following list briefly outlines the first steps in troubleshooting IP connectivity issues and Cisco IOS CEF:

- Step 1. Accurately describe the problem.
- Step 2. Scope the network topology.
- Step 3. Review the Open Systems Interconnection (OSI) model.
Verify the physical layer.
Verify the Layer 2 layer topology.
- Step 4. Verify the Address Resolution Protocol (ARP) table.
- Step 5. Verify the IP routing table.
- Step 6. Verify the CEF Forwarding Information Base (FIB) table.
- Step 7. Verify the adjacency table.
- Step 8. Conduct hardware-specific troubleshooting.

Accurately Describe the Problem

Accurately articulating your IP connectivity problem is paramount to troubleshooting effectively. An ad hoc approach to troubleshooting is usually ineffective in resolving problems. For example, you do not go to your dentist and tell him you are in pain without describing the symptoms, such as which tooth, how often, how intense, how widespread, what causes the pain, and so on. The same premise exists with IP connectivity troubleshooting. To help yourself, you need to know as much about the issue as possible.

The following questions aid you in accurately articulating your IP connectivity problem:

- Is your IP connectivity problem isolated to a single end device or multiple end devices?
- Is your IP connectivity problem isolated to a single router or Ethernet switch?
- Does your IP connectivity problem exist only on end devices, or does it affect the management CLI of routers and switches as well?
- How widespread is the problem?
- Is the problem widespread or localized to specific area of your network topology?
- Is the problem intermittent or consistent? For example, using the Internet Control Message Protocol (ICMP) ping utility in Cisco IOS and on end devices, are you getting intermittent responses to ICMP echo requests such as every other response, no responses, or inconsistent responses (one out of ten)?

- Does this issue depend on packet size? If you send ICMP echo requests at different sizes, do you consistently get all your responses or does the problem vary with packet size?
- When did the problem first occur? Were there any changes to the network at the same time the problem started occurring?

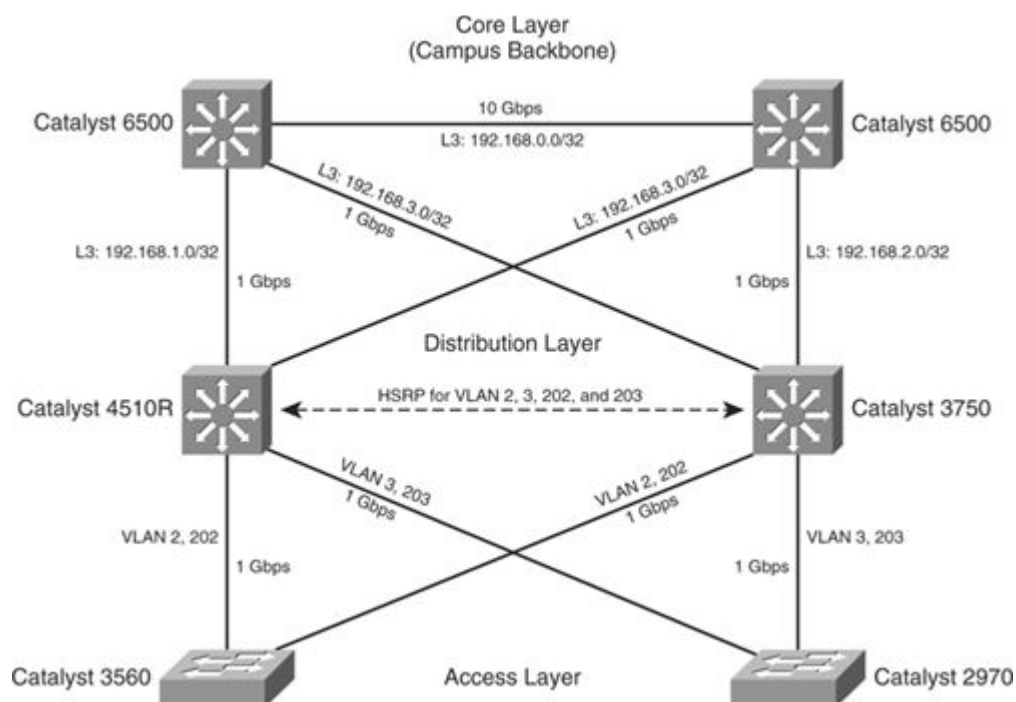
These questions aid you in articulating your IP connectivity issues. The next section describes an important next step, building the network topology.

Scoping the Network Topology

It is nearly impossible to troubleshoot any type of CEF issue or network connectivity issue without a network diagram that depicts IP addresses, IP routes, devices such as firewalls and switches, and so on. Troubleshooting IP connectivity problems without the aid of a visual topology is nearly impossible unless you can localize the issue to a specific router or switch. In large IP routing scenarios, a network topology is required to troubleshoot connectivity problems. Generally, both logical and physical topologies aid in troubleshooting. Figure 4-1 illustrates a sample physical topology.

Figure 4-1. Sample Network Topology

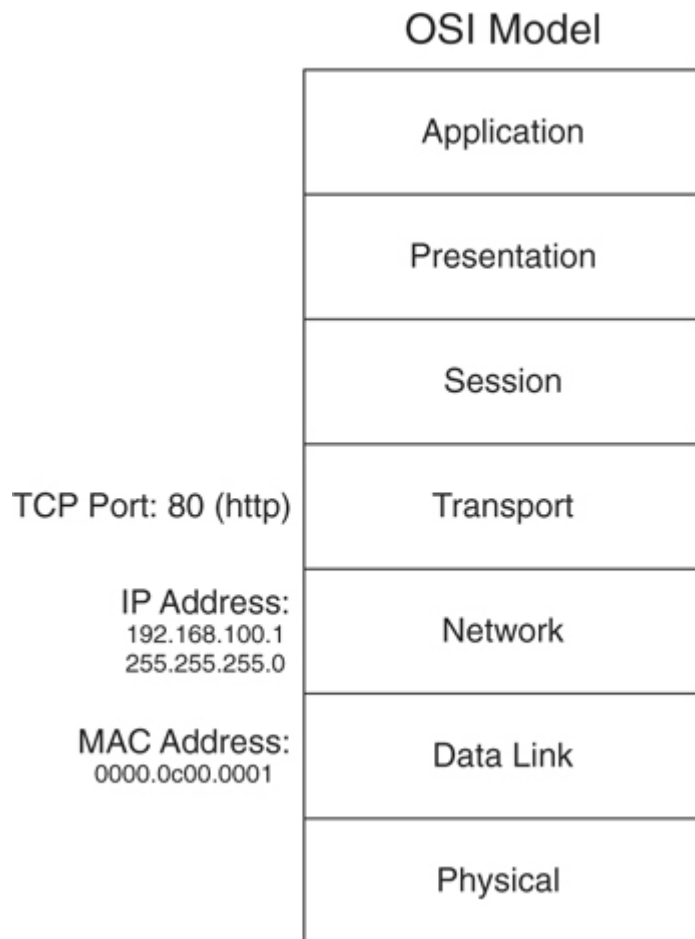
[View full size image]



Reviewing the OSI Model for Troubleshooting

The next step in troubleshooting any IP connectivity issue is to review the OSI model and verify that your issue is indeed a Layer 3 (network) issue. Figure 4-2 briefly reviews the OSI model. As a reader of this high-level technology book on CEF, we assume you have an understanding of the OSI model.

Figure 4-2. OSI Model



The following sections start from the bottom of the OSI model and review troubleshooting physical connectivity and Layer 2 issues that can affect IP connectivity and give the appearance of a CEF issue.

Troubleshooting Physical Connectivity

An IP connectivity issue might simply be a Layer 1 (physical layer) problem. For example, if you are unable to ping a network device through a router, do not assume that you are having a CEF issue. First, ensure that the host is connected and verify that the physical layer between the host and destination is not sustaining errors. Example 4-1 illustrates sample output from a show interfaces command in Cisco IOS.

Example 4-1. Verifying the Physical Layer

Code View: Scroll / Show All

```
Switch#show interfaces GigabitEthernet 3/6
```

```
GigabitEthernet3/6 is up, line protocol is up (connected)
```

```
Hardware is Gigabit Ethernet Port, address is 0010.7bfa.808d (bia 0010.7bfa.808d)
```

```
MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
```

reliability 255/255, txload 1/255, rxload 1/255

Encapsulation ARPA, loopback not set

Keepalive set (10 sec)

Full-duplex, 1000Mb/s, link type is auto, media type is 1000BaseSX

input flow-control is on, output flow-control is off

ARP type: ARPA, ARP Timeout 04:00:00

Last input 00:00:22, output never, output hang never

Last clearing of "show interface" counters 5w0d

Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0

Queueing strategy: fifo

Output queue: 0/40 (size/max)

5 minute input rate 0 bits/sec, 0 packets/sec

5 minute output rate 2000 bits/sec, 2 packets/sec

683355349 packets input, 357724908540 bytes, 0 no buffer

Received 155022 broadcasts (155022 multicast)

0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored

0 input packets with dribble condition detected

338647150 packets output, 23295655000 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 babbles, 0 late collision, 0 deferred

0 lost carrier, 0 no carrier

0 output buffer failures, 0 output buffers swapped out

In regard to the show interfaces command and verifying physical connectivity, verify that your ingress and egress interfaces are not sustaining errors such as input errors, cyclic redundancy check (CRC) errors, output errors, excessive collisions, overruns, late collisions, or output buffer failures. These types of errors can lead to intermittent or total loss of IP connectivity. Generally, physical layer issues cause intermittent connectivity if

the connection has link. Layer 1 errors can be a result of a bad cable, bad port, faulty hardware, and so on. Before proceeding with IP connectivity troubleshooting and ultimately CEF troubleshooting, you must rule out Layer 1 issues.

Troubleshooting Layer 2 Issues

The next step in troubleshooting intermittent connectivity, after physical layer (Layer 1) issues have been ruled out, is to verify that the Layer 2 topology is operating correctly. Verifying the Layer 2 topology includes, but is not limited to, verifying the spanning-tree topology, MAC address table, and Hot Standby Routing Protocol (HSRP) status in switched networks. In a point-to-point router configuration, these features are seldom used. Nevertheless, if your topology involves aggregating and connecting routers to Ethernet switches, you must check the Layer 2 topology before assuming that CEF is the root cause of an IP connectivity issue. Because CEF is found on all Layer 3 Cisco Catalyst switches, troubleshooting Layer 2 issues is necessary when troubleshooting IP connectivity issues on these platforms.

The following list articulates the most common Layer 2 features that can affect IP connectivity when not operating in a correct state:

802.1D, 802.1w, and 802.1s Spanning Tree Protocols

Hot Standby Routing Protocol (HSRP)

Virtual Router Redundancy Protocol (VRRP)

EtherChanneling (port channeling)

Compression (software- or hardware-based)

Encryption (software- or hardware-based)

Firewalls

Virtual Private LAN Services (VPLS)

Any Transport over Multiprotocol Label Switching (AToM)

Consult the list for applicable features in your network topology and refer to the Cisco.com website for more details on troubleshooting these features. Troubleshooting these features is outside the scope of this book.

Because many Cisco IOS router experts are not also Cisco IOS switch experts, router experts can find switch troubleshooting daunting. Nevertheless, you should rule out Layer 2 issues as a cause for IP connectivity and CEF issues. The most efficient way to troubleshooting Layer 2 issues is to remove redundant paths, disable as many features as possible, and isolate connectivity issues to a single host.

After the mundane Layer 1 and Layer 2 issues have been ruled out, you can finally transition to troubleshooting IP and CEF, the main focus of this book. As with any troubleshooting, you use a stepwise approach.

Verifying the ARP Table

The first step in troubleshooting IP connectivity issues and CEF from a Layer 3 perspective is to consult the ARP table for identified IP devices that are experiencing connectivity issues. When CEF cannot locate a valid adjacency for a destination prefix, it punts the packets to the CPU for ARP resolution and, in turn, completion of the adjacency.

For example, if the ARP table already lists a particular host, punting it to the process level does not trigger an ARP request. If an entry is incomplete (no response from ARP request) or incorrect in the ARP table, it is also incomplete or incorrect in the CEF adjacency table. This section covers reviewing the ARP table.

Example 4-2 illustrates sample output from the show arp command used to display the contents of the ARP table.

Example 4-2. Displaying the ARP Table in Cisco IOS

```
Router-2#show arp

Protocol Address      Age (min) Hardware Addr  Type  Interface
-----
Internet 172.18.114.250    0 0007.e978.ef03 ARPA  Vlan114
Internet 172.18.114.244   237 0004.7553.cf3c ARPA  Vlan114
Internet 172.18.114.243    0 Incomplete   ARPA  Vlan114
```

Based on the show arp command output, investigate whether the ARP table information is correct. In this example, you see three entries. The 172.18.114.243 entry is incomplete, which means that the device with the address 172.18.114.243 did not respond to the ARP request or is simply powered off. To verify whether the other listings are correct, you might need to access the end device and verify the locally configured MAC address.

Example 4-3 illustrates determining the IP address and MAC address of a Microsoft Windows XP laptop.

Example 4-3. Determining the IP Address and MAC Address of a Microsoft Windows XP Laptop

```
Code View: Scroll / Show All

C:\WINDOWS\system32>ipconfig.exe /all

Windows IP Configuration

Host Name . . . . . : test-winxp
Primary Dns Suffix . . . . . : amer.bcmsn.com
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : cisco.com
                                     cisco.com
```

Ethernet adapter Local Area Connection 2:

Connection-specific DNS Suffix . : cisco.com
Description : (text deleted) Network Connection
Physical Address. : 00-09-6B-60-15-74
Dhcp Enabled. : Yes
Autoconfiguration Enabled : Yes
IP Address. : 172.18.114.244
Subnet Mask : 255.255.0.0
Default Gateway : 172.18.114.1
DHCP Server : 10.18.0.1
DNS Servers : 10.2.2.1
Lease Obtained. : Saturday, February 12, 2005 12:08:5 PM
Lease Expires : Sunday, February 13, 2005 12:38:59 PM

Ethernet adapter Wireless Network Connection 3:

Media State : Media disconnected
Description : Cisco Systems PCI Wireless LAN Adapter
Physical Address. : 00-02-8A-3E-6D-CB

In Example 4-3, the MAC (physical) address of the host is 00-09-6B-60-15-74. This differs from the ARP table of the local router. A protocol analyzer such as Ethereal is necessary to decode frames on Ethernet to investigate why the router has an incorrect entry. Example 4-4 shows an example of troubleshooting ARP entries with the debug arp command.

Note

Because this is simply a mock setup for illustrative purposes, using a Cisco IOS debug is not an issue. Generally, in large networks, debugs such as debug arp are intrusive and should not be used in production environments.

Example 4-4. Troubleshooting Incorrect ARP Entries Using the debug arp Command

Code View: Scroll / Show All

```
Router-2#debug arp
```

```
ARP packet debugging is on
```

```
Router-2#clear arp int vlan 114
```

```
*Feb 12 10:43:23.710 UTC: IP ARP: sent req src 172.18.114.4 0008.a378.bdff,  
dst 172.18.114.250 0007.e978.ef03 Vlan114
```

```
*Feb 12 10:43:23.710 UTC: IP ARP: sent req src 172.18.114.4 0008.a378.bdff,  
dst 172.18.114.244 0004.7553.cf3c Vlan114
```

```
*Feb 12 10:43:23.718 UTC: IP ARP: rcvd rep src 172.18.114.250 0007.e978.ef03, dst  
172.18.114.4 Vlan114
```

```
*Feb 12 10:43:23.718 UTC: IP ARP: creating entry for IP address: 172.18.114.250,  
hw: 0007.e978.ef03
```

```
*Feb 12 10:43:23.718 UTC: IP ARP: rcvd rep src 172.18.114.244 0009.6B60.1574, dst  
172.18.114.4 Vlan114
```

```
*Feb 12 10:43:23.718 UTC: IP ARP: creating entry for IP address: 172.18.114.244,  
hw: 0009.6B60.1574
```

```
*Feb 12 10:43:24.124 UTC: IP ARP: rcvd rep src 172.18.114.244 0004.7553.cf3c, dst  
172.18.114.4 Vlan114
```

```
*Feb 12 10:43:24.124 UTC: IP ARP: creating entry for IP address: 172.18.114.244,  
hw: 0004.7553.cf3c
```

From the debug output, it is apparent that two devices are replying to the ARP request sent by the router: the test host and a rogue or misconfigured device. The router always populates the ARP table with the most recent ARP response. Because the rogue entry arrived second, the ARP table was incorrect. Another example of invalid ARP entries is with Frame Relay point-to-point or other types of point-to-point interfaces. Because the ARP entry is incorrect, the CEF adjacency table entry will also be incorrect.

Note

When the ping process on a Cisco router or switch running Cisco IOS attempts to send an ICMP echo to a host for which an ARP entry does not exist, the router or switch initiates an ARP request. Because the first ICMP echo is dropped after a defined period awaiting the ARP response, the first ICMP echo fails to be sent. As a result, initiating an ICMP ping with default parameters to a device for which a current ARP entry does not exist generally results in a success rate of 4/5 (80 percent) because five ICMP echoes are sent by default in Cisco IOS.

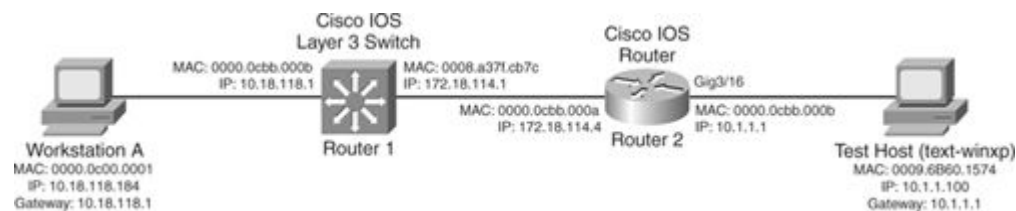
Moreover, a common IP connectivity problem associated with CEF is where a more specific route, such as a host route (/32), is learned on a different interface than the connected route (for example, a host, 192.168.1.10, connected on interface Ethernet 0/0 of Cisco IOS router). According to the routing table, 192.168.1.0/24 is learned as a directly connected route. If the router learns a more specific route, such as 192.168.1.10/32, from another interface, the router forwards the packets to the more specific route. This is a common issue; you should rule it out early in troubleshooting.

Verifying the Routing Table

The example in the previous section was based on a directly connected device. To illustrate verifying the IP routing table and subsequent CEF troubleshooting, Example 4-5 is based on a router with several routes, as illustrated in Figure 4-3. Devices connected to Router 2 in the 10.1.1.0/24 subnet are unable to reach a Secure File Transfer Protocol (SFTP) server with the address 10.18.118.184, as shown in the first steps of Example 4-5; specifically, the host 10.1.1.100 is unable to ping 172.18.118.184. For the purpose of this example, assume that all the IP routing configuration and host configurations, such as IP address and default gateways, are configured correctly. In addition, assume that our issue is strictly limited to Router 2 and not Router 1, the Layer 3 switch.

Figure 4-3. Troubleshooting IP Connectivity and Verifying the IP Routing Table

[View full size image]



Example 4-5. IP Connectivity Issue from the Host, 10.1.1.100, to the Target, 172.18.118.184, Based on Figure 4-3

```
Code View: Scroll / Show All

Host 10.1.1.100:

C:\WINDOWS\system32>ping 10.18.118.184
```

Pinging 172.18.118.184 with 32 bytes of data:

Request timed out.

Request timed out.

Request timed out.

Request timed out.

Ping statistics for 172.18.118.184:

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\WINDOWS\system32>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

Connection-specific DNS Suffix . : cisco.com

IP Address. : 10.1.1.100

Subnet Mask : 255.255.255.0

Default Gateway : 10.1.1.1

C:\WINDOWS\system32>tracert 10.18.118.183

Tracing route to 10.18.118.183 over a maximum of 30 hops

1 <10 ms <10 ms <10 ms Router-2.cisco.com [10.1.1.1]

2 * * * Request timed out.

```
3 * * * Request timed out.
4 * * * Request timed out.
5 * * * Request timed out.
6 * * * Request timed out.

!Output omitted
```

The first step in troubleshooting is to access the first-hop router, Router 2, and verify IP connectivity to the SFTP server, 10.18.118.184. Example 4-6 also confirms the route to the host, 10.18.118.184, and verifies the ARP entry for the next hop for 10.18.118.184, which is 10.18.114.1. The host entry, 10.18.118.184, is known through a static route with a metric of 0 according to the show ip route command. This is the gateway of last resort.

Example 4-6. Verifying IP Routing Table and Next-Hop Information

```
Code View: Scroll / Show All

Router-2#ping 172.18.118.184

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.18.118.184, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms

Router-2#

Router-2#show ip route 172.18.118.184

Routing entry for 172.18.118.0/24

  Known via "static", distance 1, metric 0

  Routing Descriptor Blocks:

    * 172.18.114.1

      Route metric is 0, traffic share count is 1
```

```
Router-2#show ip arp 172.18.114.1
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.18.114.1	0	0008.a37f.cb7c	ARPA	Vlan114

```
Router-2#show ip route 172.18.114.1
```

```
Routing entry for 172.18.114.0/24
```

```
Known via "connected", distance 0, metric 0 (connected, via interface)
```

```
Redistributing via ospf 1
```

```
Routing Descriptor Blocks:
```

```
* directly connected, via Ethernet0/0
```

```
Route metric is 0, traffic share count is 1
```

Although the ping from the router was successful, the next step is to verify the MAC (Hardware Addr) of the next-hop router. This step verifies that the correct ARP entry exists for the next-hop router. Example 4-7 illustrates obtaining Router 1's MAC address for the interface configured with the IP address 172.18.114.1.

Example 4-7. Verifying the MAC Address of an Interface in Cisco IOS

```
Router-1#show interfaces vlan 114 | include address
```

```
!Output omitted
```

```
Hardware is Cat6k RP Virtual Ethernet, address is 0008.a37f.cb7c (bia  
0008.a37f.cb7c)
```

```
Internet address is 172.18.114.1/24
```

```
!Output omitted
```

The next-hop router's (Router 1's) MAC address is indeed 0008.a37f.cb7c. Therefore, at this point in troubleshooting, both the routing entry and ARP entry for the next-hop router are correct.

The next step is to verify IP connectivity to the next-hop router, Router 1, from both the host, 10.1.1.100, and Router 2, as shown in Example 4-8.

Example 4-8. Verifying IP Connectivity to the Next Hop from Router 2 and the Host, 10.1.1.100

Code View: Scroll / Show All

```
Router-2#ping 172.18.114.1
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.18.114.1, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms

```
Router-2#ping
```

Protocol [ip]:

Target IP address: 172.18.114.1

Repeat count [5]: 10

Datagram size [100]: 1500

Timeout in seconds [2]:

Extended commands [n]:

Sweep range of sizes [n]:

Type escape sequence to abort.

Sending 10, 1500-byte ICMP Echos to 172.18.114.1, timeout is 2 seconds:

!!!!!!!!!!

Success rate is 100 percent (10/10), round-trip min/avg/max = 1/2/4 ms

Host 10.1.1.100:

```
C:\WINDOWS\system32>ping 172.18.114.1
```

Pinging 172.18.118.184 with 32 bytes of data:

```
Reply from 172.18.114.1: bytes=32 time=41ms TTL=253
```

```
Reply from 172.18.114.1: bytes=32 time=41ms TTL=253
```

```
Reply from 172.18.114.1: bytes=32 time=41ms TTL=253
```

```
Reply from 172.18.114.1: bytes=32 time=40ms TTL=253
```

```
Ping statistics for 172.18.114.1:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
    Minimum = 40ms, Maximum = 41ms, Average = 40ms
```

Example 4-8 also issued an ICMP ping of a larger packet size for continued verification of accessibility of the next-hop router. You must verify various packet sizes because default-sized packets of ICMP echoes can be routed and switched correctly in the network, whereas larger packets can be dropped for a multitude of reasons. These reasons include encoding errors in WAN circuits, fragmentation configuration issues, hardware issues, Virtual Private Network (VPN) misconfiguration, software features such as encryption and compression, and so on. Some network routers' security policies can also prevent the routers from responding to very high rates of ICMP packets. Therefore, you should understand your network policies when troubleshooting or installing systems. In Example 4-8, the router is indeed reachable through ICMP.

Example 4-8 proved that the next hop is reachable from both Router 2 and the host, 10.1.1.100. Therefore, at this point, the ARP and IP routing tables appear to be correct on Router 2. The next step is troubleshooting CEF (Step 6), which is discussed in the next section using the same example as that in Figure 4-3.

Sending ICMP echoes, requests, or responses from the CLI of the router is software switched, while ICMP echoes from end devices through many Cisco IOS routers and all current-generation Catalyst switches are hardware switched. The software-switching path might be correct, but the hardware-switching path might not be correct. In this example, the software-switching path on Router 2 might be correct, but the hardware-switching path might not be correct. Troubleshooting the hardware-switching path is outside the scope of this chapter. Chapter 5 discusses troubleshooting hardware-switching paths on a Cisco Catalyst 6500 platform. However, you must troubleshoot CEF from the software-switching perspective first because the hardware-switching tables are built from the software-switching tables.

The section "Troubleshooting the CEF FIB Table," later in this chapter, continues the investigation of the software-switching path by troubleshooting the Cisco IOS CEF table on Router 2 for Figure 4-3. As noted previously, because the hardware-switching path gets built from the software-switching CEF and adjacency table, you should investigate the software-switching path first.

Using IOS Ping with the Record Option to Rule Out CEF

CEF does not support all IP packet types and must process-switch specific types of packets. One such packet includes the ICMP echo with the record option. As such, you can rule out CEF as a cause of an IP connectivity problem with some certainty using the ICMP echo with record option.

For example, in Figure 4-3, sending an ICMP echo with the record option forces all routers along the path to use the process-switching method of forwarding a frame. If an ICMP echo with the record option is successful and a standard ICMP echo is not, you can assume with some certainty that CEF is indeed a cause of your IP connectivity issue somewhere along the path.

In Cisco IOS, use the ping command with extended commands option to send ICMP echoes with the record option. Example 4-9 illustrates the use of the ICMP echo with record option.

Example 4-9. Sending ICMP Packets Using the Record Option

Code View: Scroll / Show All

```
Router-2#ping ip
Target IP address: 10.18.118.184
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: loop 0
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: r
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.18.118.184, timeout is 2 seconds:
Packet has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
```

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

Reply to request 0 (1 ms). Received packet has options

Total option bytes= 40, padded length=40

Record route:

(172.18.114.4)

(10.18.118.1)

(10.18.118.184)

(172.18.114.4) <*>

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

(0.0.0.0)

End of list

!Output omitted for brevity

Note

When troubleshooting CEF, always note that certain packets, mainly those requiring special handling, are not CEF switched.

Troubleshooting the CEF FIB Table

The following sections outline basic CEF troubleshooting using a stepwise approach based on Figure 4-3 and Example 4-5. These sections move to focusing on CEF.

The steps for verifying the CEF table are as follows:

- Step 1. Verify the CEF configuration.
- Step 2. Confirm the IP CEF switching path, including using CEF accounting counters to confirm the switching path.
- Step 3. Verify CEF switching details.

Verifying the CEF Configuration

When verifying the CEF table (FIB), first verify whether CEF is indeed enabled globally and on an interface basis using the following commands:

```
show ip cef
```

```
show cef interface <module_number>/<port_number>
```

Example 4-10 illustrates how to verify that CEF is indeed enabled on both a global and an interface basis.

Example 4-10. Verifying That IP CEF Is Enabled Globally and Per-Interface

Code View: Scroll / Show All

```
Router-2#show ip cef
```

Prefix	Next Hop	Interface
0.0.0.0/0	172.18.114.1	Ethernet0/0
0.0.0.0/32	receive	
10.0.0.0/24	172.18.114.1	Ethernet0/0
10.1.1.0/24	attached	Ethernet0/1
10.1.1.0/32	receive	
10.1.1.1/32	receive	
10.1.1.100/32	10.1.1.100	Ethernet0/1

```

10.1.1.255/32  receive
10.18.118.0/24  172.18.114.1  Ethernet0/0
10.224.0.0/24  172.18.114.1  Ethernet0/0
10.225.0.0/24  172.18.114.1  Ethernet0/0
10.226.0.0/24  172.18.114.1  Ethernet0/0
165.27.1.0/24  172.18.114.1  Ethernet0/0
172.18.114.0/24  attached      Ethernet0/0
172.18.114.0/32  receive
172.18.114.1/32  172.18.114.1  Ethernet0/0
172.18.114.4/32  receive
172.18.114.5/32  172.18.114.5  Ethernet0/0
172.18.114.7/32  172.18.114.7  Ethernet0/0
172.18.114.177/32  172.18.114.177  Ethernet0/0
172.18.114.191/32  172.18.114.191  Ethernet0/0
172.18.114.214/32  172.18.114.214  Ethernet0/0
Prefix      Next Hop      Interface
172.18.114.255/32  receive
172.18.116.64/29  172.18.114.1  Ethernet0/0
192.168.100.0/24  172.18.114.1  Ethernet0/0
224.0.0.0/4      drop
224.0.0.0/24      receive
255.255.255.255/32  receive

```

If CEF is not enabled globally, the show ip cef command returns the message "%CEF not running," as shown in Example 4-11.

Example 4-11. Example of Router Not Running CEF

```
Router-2#show ip cef
```

```
%CEF not running
```

Although the previous step shows that CEF is enabled on the interfaces in respect to Figure 4-3, the show ip interface and the show cef interface interface commands confirm that CEF is enabled on a per-interface basis. Example 4-12 illustrates the show ip interface command.

Example 4-12. Verifying That CEF Is Enabled on a Per-Interface Basis Using the show ip interface Command

```
Code View: Scroll / Show All
```

```
Router-2#show ip interface ethernet 0/0  
Ethernet0/0 is up, line protocol is up  
Internet address is 172.18.114.4/24  
Broadcast address is 255.255.255.255  
Address determined by setup command  
MTU is 1500 bytes  
Helper address is not set  
Directed broadcast forwarding is disabled  
Multicast reserved groups joined: 224.0.0.5 224.0.0.6  
Outgoing access list is not set  
Inbound access list is not set  
Proxy ARP is enabled  
Security level is default  
Split horizon is enabled  
ICMP redirects are always sent  
ICMP unreachable are always sent  
ICMP mask replies are never sent  
IP fast switching is enabled  
IP fast switching on the same interface is disabled  
IP Flow switching is disabled
```

IP CEF switching is enabled
IP CEF Fast switching turbo vector
IP multicast fast switching is enabled
IP multicast distributed fast switching is disabled
IP route-cache flags are Fast, CEF
Router Discovery is disabled
IP output packet accounting is disabled
IP access violation accounting is disabled
TCP/IP header compression is disabled
RTP/IP header compression is disabled
Probe proxy name replies are disabled
Policy routing is disabled
Network address translation is disabled
Web Cache Redirect is disabled
BGP Policy Mapping is disabled

Router-2#show cef interface ethernet 0/0

Ethernet0/0 is up (if_number 2)

Corresponding hwidb fast_if_number 2

Corresponding hwidb firstsw->if_number 2

Internet address is 172.18.114.4/24

ICMP redirects are always sent

Per packet load-sharing is disabled

IP unicast RPF check is disabled

Inbound access list is not set

Outbound access list is not set

IP policy routing is disabled

BGP based policy accounting is disabled

```

Hardware idb is Ethernet0/0

Fast switching type 1, interface type 61

IP CEF switching enabled

IP CEF Feature Fast switching turbo vector

Input fast flags 0x0, Output fast flags 0x0

ifindex 1(1)

Slot 0 Slot unit 0 Unit 0 VC -1

Transmit limit accumulator 0x0 (0x0)

IP MTU 1500

```

Confirming the IP CEF Switching Path

The next step in troubleshooting the CEF table is to verify the switching path that the router in question is using. The `show interfaces stat` command displays the switching path stats on Cisco IOS routers, as shown in Example 4-13. The processor row includes process-switched (software-switched) packets, while the router cache row includes both CEF-switched and fast-switched packets. Note that high-end routers and switches that support distributed CEF display an additional row referred to as Hardware, Parallel Express Forwarding (PXF), or Distributed.

Example 4-13. Displaying Interface Switching Statistics

```

Router-2#show interfaces stat

Ethernet0/0

  Switching path  Pkts In  Chars In  Pkts Out  Chars Out
  Processor      398302  28590641  34250    3035319
  Route cache     173     81340    192     12152
  Total          398475  28671981  34442    3047471

Ethernet0/1

  Switching path  Pkts In  Chars In  Pkts Out  Chars Out

```

Processor	14614	3087533	18038	1700974
Route cache	191	12955	172	80920
Total	14805	3100488	18210	1781894

From Example 4-13, the router is using route cache for switching. The command does not tell you how much is CEF switched or how much is fast switched because route cache includes both CEF-switched and fast-switched packets. Another way to determine whether the router is fast-switching or CEF-switching packets is to view the contents of the fast-switching table using the show ip cache command. If prefix entries exist in the fast-switching table, the traffic to the destination entries are being fast switched. Example 4-14 illustrates using the show ip cache command to verify CEF switching.

Example 4-14. Viewing the Fast-Switching (IP Cache) Table

Code View: Scroll / Show All

Router-2#show interface stat

Ethernet0/0

Switching path	Pkts In	Chars In	Pkts Out	Chars Out
Processor	400445	28741844	34413	3049576
Route cache	416	128348	436	29516
Total	400861	28870192	34849	3079092

Interface Serial0/0 is disabled

Ethernet0/1

Switching path	Pkts In	Chars In	Pkts Out	Chars Out
Processor	14662	3099087	18119	1708563
Route cache	436	30581	416	127949
Total	15098	3129668	18535	1836512

Router-2#show ip cache

IP routing cache 0 entries, 0 bytes

2 adds, 2 invalidates, 0 refcounts

Minimum invalidation interval 2 seconds, maximum interval 5 seconds,

quiet interval 3 seconds, threshold 0 requests

Invalidation rate 0 in last second, 0 in last 3 seconds

Last full cache invalidation occurred 1d18h ago

Prefix/Length	Age	Interface	Next Hop
---------------	-----	-----------	----------

Router-2#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router-2(config)#no ip cef

Router-2(config)#end

Router-2#show ip cef

%CEF not running

Prefix	Next Hop	Interface
--------	----------	-----------

Router-2#show ip cache

IP routing cache 2 entries, 344 bytes

4 adds, 2 invalidates, 0 refcounts

Minimum invalidation interval 2 seconds, maximum interval 5 seconds,

quiet interval 3 seconds, threshold 0 requests

Invalidation rate 0 in last second, 0 in last 3 seconds

Last full cache invalidation occurred 1d18h ago

Prefix/Length	Age	Interface	Next Hop
---------------	-----	-----------	----------

10.1.1.100/32	00:00:10	Ethernet0/1	10.1.1.100
---------------	----------	-------------	------------

10.18.118.0/24	00:00:10	Ethernet0/0	172.18.114.1
----------------	----------	-------------	--------------

Router-2#show interfaces stat

Ethernet0/0

Switching path	Pkts In	Chars In	Pkts Out	Chars Out
Processor	400665	28758026	34436	3051672
Route cache	506	135008	525	36102
Total	401171	28893034	34961	3087774

Ethernet0/1

Switching path	Pkts In	Chars In	Pkts Out	Chars Out
Processor	14674	3101447	18131	1709787
Route cache	524	37093	505	134535
Total	15198	3138540	18636	1844322

Router-2#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router-2(config)#ip cef

Router-2(config)#end

Router-2#show ip cache

IP routing cache 0 entries, 0 bytes

5 adds, 5 invalidates, 0 refcounts

Minimum invalidation interval 2 seconds, maximum interval 5 seconds,

quiet interval 3 seconds, threshold 0 requests

Invalidation rate 0 in last second, 0 in last 3 seconds

Last full cache invalidation occurred 1d18h ago

Prefix/Length	Age	Interface	Next Hop
---------------	-----	-----------	----------

Router-2#

Example 4-14 illustrates that the route cache counters from the show interface stats command were CEF switched by the router in our example. Example 4-14 views the output of the show interface stats command, checks the fast-switching table using the show ip cache command, and then disables CEF. After disabling CEF, the fast-switching table was rechecked and entries were observed.

In reference to Figure 4-3, where the host 10.1.1.100 could not ping 10.18.118.184, the troubleshooting step of ruling in or ruling out CEF is to disable CEF switching, globally or per interface, and reattempt to send the ICMP echoes. Of course, in a production environment, disabling CEF might not be an option; however, during a change-control window or isolated situation, this troubleshooting is useful. Always consult with a Cisco Technical Assistance Center (TAC) engineer before disabling CEF.

When disabling CEF on a per-interface basis for troubleshooting, pay close attention to the ingress and egress interfaces. Generally, when disabling CEF on a per-interface basis for troubleshooting, disable CEF on both the ingress and egress interface. In a mixed-mode environment, where the ingress and egress interface use different switching configurations, refer to Table 4-1 for the resulting switching method.

Ingress Interface	Egress Interface	Resulting Switching Method
CEF	Process	CEF
Process	CEF	Fast
Process	Fast Switching	Fast Switching
CEF	Fast Switching	CEF

Based on the information in Table 4-1, CEF switching occurs on the ingress. Therefore, use the no ip route-cache cef command on the ingress interface to disable CEF. In contrast, because Cisco IOS builds a fast-switching cache entry after switching a packet, packets ingress on a process-switched interface and egress through a fast-switched interface. Therefore, use the no ip route-cache command on the egress interface to disable fast switching. Alternatively, disabling CEF on a global basis is permissible on several platforms, mostly low-end platforms that use the no ip cef command. Check the release notes and configuration guide for your specific platform when attempting to disable CEF globally or per interface.

Note

To reenabling CEF on a per-interface basis, both fast switching and CEF have to be enabled. As a result, not only is the ip route-cache cef interface configuration command required, but the ip route-cache interface configuration command is also required for CEF to function on a per-interface basis.

Furthermore, newer mid- to high-end platforms do not support disabling CEF on a global or per-interface basis. When attempting to disable CEF on a platform such as the Catalyst 6500, Cisco 12000, CRS, and Cisco 7600 that do not support disabling CEF, the CLI returns an error. Example 4-15 illustrates an example of attempting to disable CEF on a platform that does not support disabling CEF.

Example 4-15. Attempting to Disable CEF on a Platform Such as the Catalyst 4500 That Does Not Support Disabling CEF

```
Catalyst4500#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Catalyst4500(config)#no ip cef
%Cannot disable CEF on this platform
```

Using CEF Accounting Counters to Confirm the Switching Path

Cisco IOS supports CEF accounting, with specific limitations pertaining to hardware-switched traffic. However, with software switching, CEF supports an accounting option for packet and byte counters.

When troubleshooting CEF, you can view the packet and byte counters on a per-prefix basis. To enable CEF accounting, use the `ip cef accounting per-prefix` global configuration command. Simply use the `show ip cef prefix` command to display the counters for a specific prefix. Example 4-16 illustrates the use of CEF accounting.

Example 4-16. CEF Accounting

```
Router-2#configure terminal
Router-2(config)#ip cef accounting per-prefix
Router-2(config)#end
Router-2#show ip cef 10.18.118.84
10.18.118.0/24, version 53, epoch 0, cached adjacency 172.18.114.1
240 packets, 14400 bytes
  via 172.18.114.1, Ethernet0/0, 0 dependencies
    next hop 172.18.114.1, Ethernet0/0
    valid cached adjacency
```

Note

When you enable network accounting for dCEF from global configuration mode, accounting information grouped by IP prefix is not sent to the route processor (RP) for viewing through the `show ip cef` command. However, the accounting information is collected by dCEF processes on the line card. In this situation, use the `show cef linecard` command to view dCEF statistics.

Verifying the CEF Switching Details

In most production environments, you usually cannot disable CEF, even on a per-interface basis, during normal production. This is because in many production networks, traffic rates exceed the software-switching capabilities of the router, and disabling CEF forces software switching. Therefore, disabling CEF to troubleshoot CEF is not always an option. The next step in these situations is to verify the CEF table information.

Example 4-17 illustrates the use of the `show ip cef` command to gather details about the CEF table on a per-entry basis. The values chosen for this example match those for the troubleshooting example.

Example 4-17. Gathering CEF Table Details

```
Router-2#show ip cef 10.18.118.184 detail
10.18.118.0/24, version 23, cached adjacency 172.18.114.1
0 packets, 0 bytes
via 172.18.114.1, Ethernet0/0, 0 dependencies
    next hop 172.18.114.1, Ethernet0/0
    valid cached adjacency

Router-2#show ip cef 172.18.114.1 detail
172.18.114.1/32, version 17, cached adjacency 172.18.114.1
0 packets, 0 bytes
via 172.18.114.1, Ethernet0/0, 1 dependency
    next hop 172.18.114.1, Ethernet0/0
    valid cached adjacency
```

From the output shown in Example 4-17, the host 10.18.118.184 next hop is 172.18.114.1 and is a valid cached adjacency through Ethernet0/0. The first step is to verify this output against Figure 4-3 and the `show ip route` command from Example 4-5. The details are correct. The next step is to verify the next-hop CEF entry for 10.18.118.184, which is 172.18.114.1. Again, this information appears to be correct based on Figure 4-3. If this information was not correct, the next step is to either disable CEF and test connectivity or open a Cisco TAC case.

Another possible reason for packet loss with CEF is CEF drop adjacency. CEF drop adjacencies define hardware-switched drops for prefixes. CEF drop adjacencies allow dropping frames in hardware rather than punting every frame for software switching. This is an effective method in preventing denial of service (DoS) attacks and high CPU usage due the CPU processing excessive drops. If a CEF drop adjacency exists, it is generally because of one of the following reasons:

Unsupported features.

Packets destined to prefixes associated with punt adjacencies that exceed a predefined rate. Punting rate limiters exist to prevent DoS attacks and high CPU usage caused by excessive punts.

Unresolved or missing FIB entry.

Unsupported frame type.

Example 4-18 illustrates a sample output from the show cef drop command. In respect to Figure 4-3 and Example 4-5, attempting to relate the CEF drop counters to the ICMP packets can yield additional details about why the packets are being dropped. However, in a production environment, it is difficult to correlate the ICMP packet loss to show cef drop counters because production environments generally have multiple flows passing traffic simultaneously. Nonetheless, if the show cef drop counters remain 0 during the ping tests, you can rule out the notion that the ping failed because of CEF drop adjacencies. In later code, the show ip cef switching statistics command gives detailed information about why a drop occurs and replaces the show cef not command.

Example 4-18. show cef drop Command Example

```
Code View: Scroll / Show All
Router-2#show cef drop
CEF Drop Statistics
Slot  Encap_fail  Unresolved  Unsupported  No_route  No_adj  ChkSum_Err
RP      3      0      0      0      0      0
```

Table 4-2 defines the fields associated with the show cef drop command.

Field	Description
Slot	Refers to the slot for the respective ingress packet counts. For Cisco IOS routers that do not support dCEF, this value is always RP.
Encap_fail	Indicates the number of packets dropped after exceeding the limit for packets punted to the processor because of missing adjacency information such as an unresolved ARP request. Note that CEF throttles packets punted to the process level at a rate of one packet per second to aid in susceptibility of DoS attacks.
Unresolved	Indicates the number of packets dropped because of an unresolved prefix in the FIB table.
Unsupported	Indicates the number of packets fast-dropped by CEF (drop adjacency) because of an unsupported feature.

Table 4-2. show cef drop Field Descriptions

Field	Description
No_route	Indicates the number of packets dropped because of a missing prefix in the FIB table.
No_adj	Indicates the number of packets dropped because of incomplete adjacency.
Chksum_Err	Indicates the number of IP version 4 (IPv4) packets received with a checksum error.

In controlled environments, preferably in nonproduction environments, use debugs to troubleshoot CEF. If you can narrow your issue to a specific host or subnet and the router or switch under investigation is logging drops or receives, you can use debugs that are limited to specific destinations to troubleshoot the issue. Limiting debugs is done by limiting debug output to specific IP sources or destinations configured in an access list. Example 4-19 illustrates an example of configuring an access control list (ACL) to limit the output of a CEF debug to a specific destination.

Example 4-19. Controlling Debug Output Defined by an ACL

Code View: Scroll / Show All

```
Router-2#configure terminal
```

Enter configuration commands, one per line. End with CNTL/Z.

```
Router-2(config)#access-list 10 permit 10.1.1.1
```

```
Router-2(config)#end
```

```
Router-2#debug ip cef drop 10
```

IP CEF drops debugging is on for access list 10

```
Router-2#debug ip cef receive 10
```

IP CEF received packets debugging is on for access list 10

```
*Mar 7 23:27:29.681 UTC: CEF-receive: Receive packet for 10.1.1.1
```

```
*Mar 7 23:27:29.681 UTC: CEF-Receive: Packet for 10.1.1.1 -- receive
```

```
*Mar 7 23:27:29.681 UTC: IP-CEF: Receive packet for 10.1.1.1 (process switch)
```

Based on the example in Figure 4-3 and Example 4-5, the debug output illustrates the router processing "receive" packets destined for its own interface. Receive packets include packets punted by CEF for software switching. Drop packets appear in the debug output in the same manner. As a result, this debug, used in controlled environments, can yield additional information helpful in troubleshooting.

Verifying the Adjacency Table

After verifying the CEF FIB table, the next step in troubleshooting CEF is to verify the adjacency table. The adjacency table contains the rewrite information that CEF uses to switch packets. Verifying that the rewrite information is correct is an important step in troubleshooting CEF operation.

The four commands that provide different hierarchical levels of information for the adjacency table are show adjacency, show adjacency summary, show adjacency detail, and show adjacency internal. Example 4-20 shows examples of these commands, respectively.

Example 4-20. Viewing Adjacency Table Details

```
Code View: Scroll / Show All

Router-2#show adjacency

Protocol Interface      Address
IP    Ethernet0/1        10.1.1.100(5)

!Output omitted for brevity

IP    Ethernet0/0        172.18.114.1(23)

Router-2#show adjacency summary

Adjacency Table has 6 adjacencies

Table epoch: 0 (6 entries at this epoch)

Interface      Adjacency Count
Ethernet0/0    5
Ethernet0/1    1

Router-2#show adjacency detail

Protocol Interface      Address
IP    Ethernet0/1        10.1.1.100(5)
```

561 packets, 41514 bytes

0007E905156C00503EFA37810800

ARP 03:53:37

Epoch: 0

!Output omitted for brevity

IP Ethernet0/0 172.18.114.1(23)

581 packets, 43012 bytes

Protocol Interface Address

0008A37FCB7C00503EFA37800800

ARP 04:02:59

Epoch: 0

Router-2#show adjacency internal

Protocol Interface Address

!Output omitted for brevity

IP Ethernet0/0 172.18.114.1(23)

280 packets, 20738 bytes

0008A37FCB7C00503EFA37800800

ARP 04:02:58

Epoch: 0

Fast adjacency disabled

IP redirect enabled

IP mtu 1500 (0x0)

Fixup disabled

Adjacency pointer 0x816BFC20, refCount 23

Connection Id 0x000000

Bucket 205

IP Ethernet0/1 10.1.1.100(5)

```

269 packets, 19906 bytes

0007E905156C00503EFA37810800

ARP    03:58:29

Epoch: 0

Fast adjacency disabled

IP redirect enabled

IP mtu 1500 (0x0)

Fixup disabled

Adjacency pointer 0x816BFAD0, refCount 5

Connection Id 0x000000

Bucket 110

!Output omitted for brevity

```

Table 4-3 describes the most significant fields from the show adjacency commands in Example 4-20 for the purpose of troubleshooting. Verifying the information against the show ip route and show arp commands is necessary in verifying CEF consistency. If the values are not correct, disable CEF as a workaround and open a Cisco TAC case.

Table 4-3. show adjacency Command Field Descriptions	
Field	Description
172.18.114.1(23)	The value in parentheses, 23, refers to the number of times a FIB entry points to an adjacency entry (refCount). Numerous system entries are not shown in the output of the show ip cef command. In the hardware used for this example, a minimum of five references per IP address existed. As a result, four additional FIB routing entries point to the IP address.
0008A37FCB7C0008A 378BDFF0800	The first 12 characters, 0008A37FCB7C, are the MAC address associated with the destination next-hop interface (destination MAC address rewrite). The next 12 characters represent the MAC address of the source interface of the packet (source MAC address rewrite). The last four characters represent the well-known Ethertype value 0x0800 for IP for Advanced Research Projects Agency (ARPA) encapsulation, the default among Ethernet vendors.

Table 4-3. show adjacency Command Field Descriptions

Field	Description
ARP 04:02:58	The ARP value indicates that the entry was learned through the ARP process. The timestamp indicates the time remaining before the entry times out. The default ARP timeout is 4 hours.
Fast adjacency disabled	An FIB entry caches an adjacency for a next-hop interface when not doing load sharing over multiple active paths. A fast adjacency increases the switching speed of packets.

In Figure 4-3 and Example 4-5, the destination MAC address rewrite information from the show adjacency detail command in Example 4-20, 0008A37FCB7C, must match the MAC address from the show arp command. Otherwise, if the MAC address did not match, an inconsistency issue exists between the ARP table and adjacency table that needs to be investigated with the Cisco TAC.

Hardware-Specific Troubleshooting

If you are troubleshooting CEF on a Cisco IOS router that is only performing software switching and are unable to find any issues based on the previous sections, the next step is to open a Cisco TAC case and include relevant information. However, if you are troubleshooting a platform that supports dCEF or hardware switching, more analysis is needed.

Chapter 5 provides information on troubleshooting hardware switching on a Cisco Catalyst 6500. Example 4-21 illustrates performing troubleshooting of dCEF by examining the hardware-switching adjacency table on a Catalyst 4500 switch.

Example 4-21. Viewing a Hardware-Switching Adjacency Table

Code View: Scroll / Show All

```
Router-2#show platform hardware ip adjacency host ip 172.18.114.1
```

```
32757: src: 00:08:A3:78:BD:FF dst: 00:08:A3:7F:CB:7C
```

```
lbc: 0 vlan: 114 port: 248 (Gi1/1) sifact: FwdToCpu
```

```
ifaid: 4086 packets: 290226195 bytes: 249761629484085 size: 1 refs: 3
```

```
age: 2856047.074sec umda: flood
```

```
vlanId: 114 single: true shar: true
```

```
ifa: (4086) int: Vl114 (vlan 114) 172.18.114.1 00:08:A3:7F:CB:7C normal
```

Troubleshooting Punt Adjacencies

In [Example 4-22](#), suppose that the `show ip cef` command output yielded the following as the result of a configuration change, such as routing the traffic over a tunnel interface.

Example 4-22. Determining Whether the FIB Prefix Points to Punt Adjacency

```
Router-2#show ip cef 10.18.118.1
10.18.118.0/24, version 31, epoch 0
0 packets, 0 bytes
  via 192.168.1.1, 0 dependencies, recursive
    next hop 192.168.1.1, Tunnel1 via 192.168.1.0/24
    valid punt adjacency
```

The CEF table output indicates that the entry is a valid punt adjacency. The term punt in Cisco IOS refers to sending a packet to the next-level switching process. By default, Cisco IOS always uses the fastest switching method possible; however, the fastest switching method usually does not support advanced features such as Network Address Translation (NAT) and policy-based routing (PBR) early in the product life cycle. Therefore, to handle these cases effectively, Cisco IOS punts the packet to the next-level switch method that can switch the frame. The following list illustrates the typical switching method hierarchy, with the fastest, most effective switching method listed first:

- Hardware-based dCEF (hardware-based switching on line cards)
- Hardware-based CEF (hardware-based, centralized forwarding typically found on Cisco Catalyst switches)
- PXF switching
- Software-based CEF
- Software-based fast switching
- Software-based process switching

Generally, the fastest and most effective switching methods generally lag software-based CEF in feature support. Therefore, when using new and unique features, Cisco IOS generally supports the feature in software first in low- to mid-range routers and switches. High-end routers and switches only support advanced features in hardware because the throughput required for the high-end routers and switches exceeds the capability of software-based switching methods. Use the `show cef not-cef-switched` command to view packets that are not CEF switched. In later code, the `show ip cef switching statistics` command gives detailed information of why a pass occurs and replaces the `show cef not` command. [Example 4-23](#) illustrates an example of the `show cef not-cef-switched` command followed by [Table 4-4](#), which illustrates descriptions for each of fields.

Example 4-23. Sample Output from the `show cef not-cef-switched` Command

Code View: [Scroll](#) / [Show All](#)

```
Router-2#show cef not-cef-switched
CEF Packets passed on to next switching layer
Slot  No_adj No_encap Unsupp'ted Redirect  Receive  Options  Access
Frag
RP      4      0      0      34      2920      0      0
0
```

Table 4-4. show cef not-cef-switched Command Field Descriptions

Field	Description
Slot	Refers to the slot for the respective ingress packet counts. For Cisco IOS routers that do not support dCEF, this value is always RP.
No_adj	Indicates the number of packets dropped because of incomplete adjacency.
No_encap	Indicates the number of packets sent to the processor for ARP resolution.
Unsup'ted	Indicates the number of packets fast-dropped by CEF (drop adjacency) because of an unsupported feature.
Redirect	Indicates the number of packets requiring ICMP redirect by process switching.
Receive	Indicates the number of packets ultimately destined to the router, or packets destined to a tunnel endpoint on the router. If the decapsulated tunnel packet is IP, the packet is CEF switched. Otherwise, packets are process switched.
Options	Indicates the number of packets with options. Packets with IP options are process switched.
Access	Indicates the number of packets punted because of an access list failure.
Frag	Indicates the number of packets punted because of fragmentation failure.
MTU	Indicates the number of packets punted because of maximum transmission unit (MTU) failure. Note: This field is not supported for IPv4 packets.

As hardware-switching components evolve, more features are being supported by CEF and dCEF. Unfortunately, most legacy software features are not supported by CEF or dCEF. Nevertheless, when troubleshooting CEF, you should note whether a software feature such as NAT, PBR, or accounting supports CEF. To verify such features, check the platform-specific release notes, data sheets, and configuration guides.

Understanding CEF Error Messages

With current-generation Cisco IOS routers and Catalyst switches, some error messages pertaining to CEF are platform specific. As a result, you need to understand platform differences when troubleshooting CEF error messages. Nevertheless, [Table 4-5](#) lists the general CEF EXEC error messages applicable to all Cisco IOS routers and Catalyst switches when configuring CEF and CEF features.

Table 4-5. General CEF EXEC Error Messages

Error Message	Troubleshooting Action
Must enable IP routing first.	When enabling CEF, IP routing must be enabled first.
Cannot disable CEF on this platform.	When disabling CEF, some platforms, such as the Catalyst 6500, do not support disabling CEF.
CEF is not enabled.	When attempting to configure a global configuration command that requires CEF, CEF must first be enabled.
CEF not enabled on this interface. Enable first.	When attempting an interface configuration that requires CEF, CEF must be enabled first.

A useful tool when working with CEF error messages is the Error Message Decoder on Cisco.com found at the following website (you must be a registered Cisco.com user):

Troubleshooting Commands Reference

Table 4-6 reviews the most important commands discussed and illustrated in this chapter.

Table 4-6. Important CEF Troubleshooting Commands

Command	Description
debug ip cef	Enables debugging with IP CEF from the Cisco IOS CLI. Because this debug is intrusive, the recommended procedure is to use this debug option with optional parameters including ACLs to limit output. Furthermore, you should use this debug command under the supervision of a Cisco TAC engineer.
show adjacency detail	Displays the IP CEF Layer 2 rewrite information and statistical information from the perspective of the software-switching path.
show arp	Displays the ARP table contents. This command is the first step in troubleshooting adjacency issues because the adjacency table is built from the ARP table.
show ip cef	Displays the IP CEF information from the software-switching path perspective. This command displays the prefix, next hop, and next-hop interface (outgoing interface) for verification. This command does not clearly illustrate the information contained through the hardware-switching path.
show ip route	Displays the IP routing table. This command is the first step in troubleshooting IP routing or IP CEF issues because the CEF tables are built from the IP routing table from a high-level

Table 4-6. Important CEF Troubleshooting Commands	
Command	Description
	perspective.

The following checklist reviews the important points of troubleshooting a software-based CEF issue:

- Never assume that you are experiencing a Cisco IOS CEF issue; always troubleshoot from Layer 1 (physical layer) up.
- In terms of troubleshooting CEF, verify the IP routing table and ARP table as a first step, because the CEF tables are built from those tables.
- In Cisco IOS, use the show interface command to verify physical layer connectivity as the initial step in troubleshooting any IP connectivity issue.
- Using the ICMP echo command ping from the Cisco IOS CLI always initiates echoes using the software-switching path.
- After verifying the ARP and IP routing table in Cisco IOS, verify the software-switching path for CEF by verifying the rewrite and next-hop information from the show ip cef detail and show adjacency commands.
- In controlled environments on select platforms that do not support dCEF, disabling CEF is an available troubleshooting option.
- Most mid- to high-end routers and all Catalyst switches do not support disabling CEF because CEF is required for normal operation.
- To determine why CEF is dropping ingress packets, use the show cef drop command.
- When troubleshooting CEF punt adjacencies, use the show cef not-cef-switched command to determine which CEF drop category the drops fall into.

Part II: CEF Case Studies

Chapter 5 Understanding Packet Switching on the Cisco Catalyst 6500 Supervisor 720

Chapter 6 Load Sharing with CEF

Chapter 7 Understanding CEF in an MPLS VPN Environment

Chapter 5. Understanding Packet Switching on the Cisco Catalyst 6500 Supervisor 720

This chapter describes the following topics:

- [CEF switching architecture on the Cisco Catalyst 6500](#)
- [Troubleshooting CEF on the Catalyst 6500 SUP720 platforms](#)

The Cisco Catalyst 6500 series of switches are some of the best-selling Cisco products because of their flexibility and capability for hardware switching. The Supervisor 720 (SUP720) is one of the latest supervisors available for the Catalyst 6500 series of switches. The SUP720 offers higher performance than its predecessors, with its integrated fabric and other hardware feature support. This chapter briefly explains how the SUP720 uses hardware Cisco Express Forwarding (CEF) switching and how to troubleshoot problems related to forwarding. It does not include detailed information related to the architecture of the Catalyst 6500 or SUP720. Further information on the architecture resides in the resources listed in the "[References](#)" section at the end of this chapter.

CEF Switching Architecture on the Cisco Catalyst 6500

On the SUP720, Cisco Catalyst 6500 uses CEF, which cannot be disabled, to provide hardware efficiency. Instead of performing switching in software as previous Cisco routers did, the Cisco Catalyst 6500 performs switching in hardware using CEF. You need to understand this difference when troubleshooting issues involving a SUP720.

Understanding Software-Based CEF and Hardware-Based CEF

Hardware-based CEF packet switching leverages the existing data structures built by CEF in software and then extends the capability to hardware by programming specialized hardware memories with information that the forwarding ASICs can use to quickly move packets for improved performance. In software-based CEF, the router builds the Forwarding Information Base (FIB) and adjacency tables based on the routing and Address Resolution Protocol (ARP) tables. The Catalyst 6500 switch stores this information in hardware high-speed table memory and uses efficient search algorithms for table lookups. In the Catalyst 6500, these high-speed memory tables are Ternary Content Addressable Memory (TCAM) tables. TCAM uses a combination of value matching with the ability to mask for a particular result. In this case, the result is a pointer to an entry in the hardware adjacency table that contains the next-hop MAC rewrite information.

The SUP720 lists FIB entries in the TCAM table from the most specific to the least specific entry. Therefore, the TCAM lists all entries with a subnet mask of 32 (host entry) for a particular prefix first. When the FIB and adjacency tables in the TCAM are full, the SUP720 punts traffic for the unfound destination/entry to the software-switching engine (route processor, or RP) to be handled in software. The SUP720 has an integrated RP and switch processor (SP) on the Multilayer Switch Feature Card 3 (MSFC3). The SP handles typical Layer 2 functions such as Spanning Tree and VLAN Trunking. The RP handles typical Layer 3 functions such as Enhanced Interior Gateway Routing Protocol (EIGRP) and routing.

The SUP720 also has a Policy Feature Card 3 (PFC3) to support hardware-based forwarding and features such as routing, multicast packet replication, and access control lists (ACLs). Multiple PFC3s, such as the PFC3B and the PFC3BXL, are available. On a SUP720 with a PFC3BXL, the FIB TCAM size and the adjacency table size

is 1 million entries. The type of features enabled can affect the number of TCAM entries that the SUP720 uses to store information. Each IP version 4 (IPv4) destination prefix known takes one entry in the TCAM. Therefore, the PFC3BXL version supports a large number of IPv4 routes and adjacencies that can be present in a large enterprise or service provider network. However, with IPv6 and ingress multicast routes, each destination takes two entries in the TCAM. Refer to Cisco.com for the latest information related to whether a feature impacts the total number of entries stored in TCAM.

The SUP720 pushes software CEF information down to the hardware ASICs, which can handle the packet forwarding faster. The RP on the SUP720 sends the information to the SP to program the TCAM and the hardware ASICs.

Hardware FIB lookup occurs based on the longest matching destination prefix. A match in the hardware FIB based on the destination prefix returns adjacency information that contains the packet rewrite information for the correct outbound interface. Then, forwarding of the packet occurs to the next hop in hardware. Note that before the Cisco Catalyst 6500 passes the packet outbound, it also validates security and quality of service (QoS) policies to confirm that the packet is legal and should not be dropped instead.

Centralized and Distributed Switching

The Cisco Catalyst 6500 supports centralized switching or distributed switching. Centralized switching occurs when the switch processor on the SUP720 makes forwarding decisions in hardware. All frames must pass through the centralized SP engine by the switch fabric or bus. Bus and crossbar switching modes use the centralized SP.

With use of the Distributed Forwarding Cards (DFCs) on the line modules, distributed switching can occur. With distributed switching, forwarding occurs locally on the line modules that bypass the centralized switching engine on the supervisor engine, and packets are forwarded directly between two ports or across the switch fabric.

If traffic is ingress on a card that is DFC equipped, the DFC performs the hardware switching for that line card. Therefore, you have to check the hardware CEF entries on the DFC itself when troubleshooting. You can use the `show fabric switching-mode` command to check whether a line card is DFC equipped. If the output says "DCEF" (indicating distributed CEF [dCEF]) under switching mode, it has a DFC. [Example 5-1](#) shows output from a SUP720 where Line Cards 2 and 5 have DFC modules installed.

Example 5-1. Verifying dCEF Modules

```
Cat6500-SUP720#show fabric switching-mode
Global switching mode is Compact
dCEF mode is not enforced for system to operate
Fabric module is not required for system to operate
Modules are allowed to operate in bus mode
Truncated mode is allowed, due to presence of DFC, aCEF720 module

Module Slot      Switching Mode
-----
1              Crossbar
2              dCEF
5              dCEF
6              Crossbar
9              Crossbar
```

Troubleshooting CEF on the Catalyst 6500 SUP720 Platforms

This section walks you through an example of troubleshooting a simple network connectivity issue where a SUP720 is in the path. Because of the intricate dependencies for hardware-based CEF, you must take a logical approach when troubleshooting. Typically, when troubleshooting network problems, as specified in [Chapter 4](#), "Basic IP Connectivity and CEF Troubleshooting," you start with the lower level of the OSI model (physical verification), then Layer 2, and then Layer 3. The same is true in this case also. After confirming basic items such as that the interface is available, you can check whether something is wrong with CEF entries. You verify

from the software perspective on the RP first and then go to the SP and further to the DFC if necessary to check CEF consistency.

Simple Checking of Connectivity

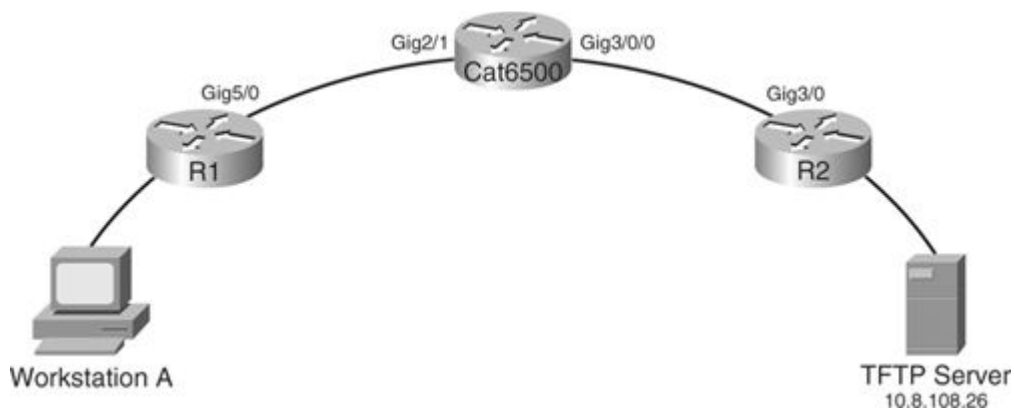
Note that the SUP720 cannot forward all packets in hardware by CEF. For example, packets with IP options set, packets with TTL=1, or packets that require fragmentation cannot be forwarded in hardware. In addition, the SUP720 cannot handle certain features such as Internetwork Packet Exchange (IPX) at the hardware level. Refer to Cisco.com or your Cisco account team for up-to-date information related to which features the SUP720 supports in hardware.

If you are unsure whether the issue is with hardware entry on the PFC3 or with the software information on the RP, do a simple extended ping test with the record bit set to confirm from the remote.

In [Figure 5-1](#), Router R1 is trying to reach Trivial File Transfer Protocol (TFTP) server 10.8.108.26 through a Cisco Catalyst 6500 with a SUP720 module. By doing an extended ping from Router R1 and setting the record bit in the ping, it forces the packet to be software-switched through the RP on the SUP720. If the extended ping works but a regular ping does not, something is most likely wrong with the hardware entries on the SUP720 device. Note that you can also do the extended ping from Workstation A to the TFTP server to test.

Figure 5-1. R1 Trying to Reach a TFTP Server

[\[View full size image\]](#)



[Example 5-2](#) shows a ping with the record option working correctly when the regular ping does not work.

Example 5-2. Example of Extended Ping Option to Check Connectivity

Code View: [Scroll](#) / [Show All](#)

```
R1#ping 10.8.108.26

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.8.108.26, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
6500-2#ping
Protocol [ip]:
Target IP address: 10.8.108.26
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]:
```



```

Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: record
Number of hops [ 9 ]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.8.108.26, timeout is 2 seconds:
Packet has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)

Reply to request 0 (1 ms). Received packet has options
Total option bytes= 40, padded length=40
Record route:
(10.9.249.139)
170-gw-test (10.0.254.78)
tftpserver (10.8.108.26)
gw-vlan (10.0.254.77)
gw-vlan4 (10.9.255.254)
(10.9.249.139) <*>
(0.0.0.0)
(0.0.0.0)
End of list

! Output omitted for brevity

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms

```

Systematic Checking of Connectivity

After performing this quick check, a more systematic way to check the tables and routing/switching information is to check the software routing, ARP, and CEF tables on the RP and then check the entries on the SP. They should match the information on the RP. If an inconsistency exists, you might have a problem with the hardware CEF and hardware forwarding. Because of the dependency of data, you must troubleshoot in a logical order. For example, hardware CEF entries depend on the software CEF entries, so you should check the software entries first.

The following list briefly outlines the steps for troubleshooting IP connectivity issues involving Cisco hardware-based CEF:

- Step 1.** Check the Cisco IOS routing table for the next-hop address. Examine the software information located on the RP. This verifies whether the next hop or hops are as expected to reach a particular destination according to the routing table.

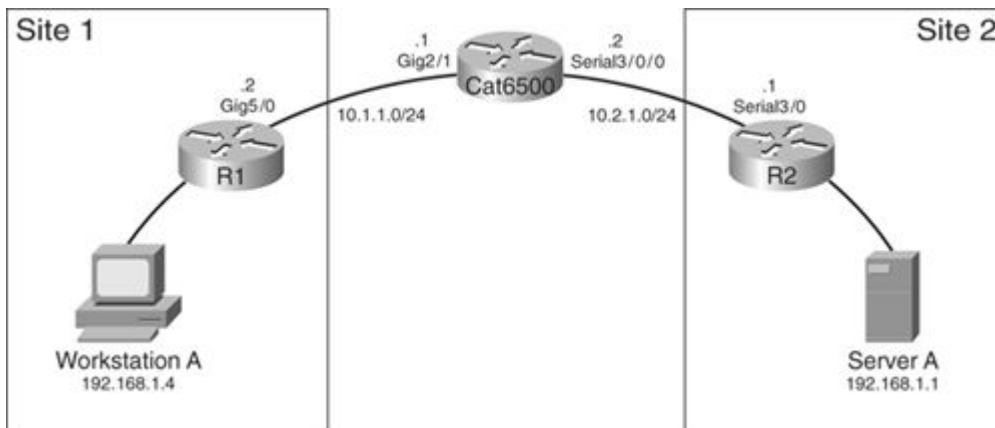
- Step 2.** Check the Cisco IOS ARP table for the MAC address for the IP next hop.
- Step 3.** Check the Cisco IOS FIB table for the next-hop address. The next-hop information in the show ip cef <destination network> output should match the next hop in the routing table information.
- Step 4.** Check the Cisco IOS adjacency table for the next hop's MAC address. Verify the adjacency information to make sure that it matches the ARP data.
- Step 5.** Check the Cisco IOS FIB and adjacency information on the SP by logging in to the SP with the remote login module command to verify software CEF information that is used to program the centralized TCAM.
- Step 6.** Check the programmed hardware FIB and adjacency information in the TCAM.
- Step 7.** Check the DFC CEF information if DFCs are installed to confirm that it matches the other data.

Table 5-1 lists some helpful commands for troubleshooting connectivity issues on the SUP720.

Table 5-1. Helpful Troubleshooting Commands for SUP720	
Show Command	Function
show ip route <prefix>	Verifies software routing entry
show ip cef <prefix> detail	Verifies software CEF entry
show mls cef <network> detail	Shows hardware forwarding information
show mls cef adjacency entry <index> detail	Shows hardware adjacency information
show vlan internal usage	Shows internal VLAN mapping and outgoing interface
show mls cef summary	Shows total number of routes known in hardware and protocol breakdown
show mls statistics	Shows how many packets were hardware forwarded and whether any errors were encountered

In Figure 5-2, consider that traffic is not passing correctly to reach 192.168.1.1 at site 2 through a Cisco Catalyst 6500 with a SUP720. Troubleshooting should start by verifying that the software information is correct. By checking the routing table on the RP, a route to 192.168.1.1/32 is known.

Figure 5-2. Troubleshooting Hardware CEF
[\[View full size image\]](#)



In [Example 5-3](#), the `show ip route` command verifies the routing table for a prefix.

Example 5-3. Step 1: Verifying the Routing Table for a Particular Prefix

```

6500#show ip route 192.168.1.1
Routing entry for 192.168.1.1/32
  Known via "ospf 1", distance 110, metric 68, type intra area
  Last update from 10.2.1.1 on Serial3/0/0, 1w3d ago
  Routing Descriptor Blocks:
    * 10.2.1.1, from 192.168.1.1, 1w3d ago, via Serial3/0/0
      Route metric is 68, traffic share count is 1
  
```

Because this is a point-to-point connection, no ARP information is stored. Therefore, skip Step 2. However, you can check to make sure that FIB information matches the routing information, as seen in [Example 5-4](#). Confirm that the next-hop address is the same and that the outbound interface is correct.

Example 5-4. Step 3: Verifying CEF Information

```

6500#show ip cef 192.168.1.1
192.168.1.1/32, version 52, epoch 0, cached adjacency to Serial3/0/0
0 packets, 0 bytes
  via 10.2.1.1, Serial3/0/0, 0 dependencies
  next hop 10.2.1.1, Serial3/0/0
  valid cached adjacency
  
```

You can check to make sure that the adjacency shows a point-to-point connection through Serial 3/0/0 in [Example 5-5](#).

Example 5-5. Step 4: Verifying Adjacency Information

```

6500#show adjacency detail | begin Serial3/0/0
IP          Serial3/0/0          point2point(17)
              664780 packets, 800395120 bytes
              0F000800
              CEF    expires: 00:02:29
                   refresh: 00:00:29
              Epoch: 0
!output omitted for brevity
  
```

[Example 5-6](#) shows an adjacency for a non-point-to-point interface. Note the difference in the MAC address rewrite portion. The MAC information is a joining of the destination MAC address and the source MAC address.

Example 5-6. Verifying Adjacency Information for a Non-Point-to-Point Interface

```
sup720#show adjacency detail
Protocol Interface Address
IP FastEthernet2/1 10.1.1.1 (7)
5 packets, 590 bytes
00E01EF2C44000115D99A0000800
FIB LC 00:00:00
Epoch: 0

!output omitted for brevity
```

If the software information is correct and consistent (Steps 1 – 5), now review the hardware information. Remember that on a Cisco Catalyst 6500, switching occurs at the hardware level. First, verify that the downloaded FIB information is correct. Use the `show mls cef <destination>` command or the `show mls cef lookup <destination> detail` command to look at the adjacency information. If the information does not exist or is inconsistent, a communication problem exists between the RP and the SP. Issuing the `clear cef linecard <slot>` command causes the RP to refresh the information downloaded to the SP. If an inconsistent condition exists and a refresh helps the communication problem, this is most likely a software bug.

Next, verify the programmed hardware information on the SP for the example in [Figure 5-2](#). In this case, traffic is coming in on non-DFC interfaces and will be handled by the SP. Looking at the information in [Example 5-7](#), you can see that a /32 entry exists for 192.168.1.1. After grabbing the adjacency index pointer value, you can look up this entry in detail.

Example 5-7. Identifying the Adjacency Index Pointer

Code View: [Scroll](#) / [Show All](#)

```
6500#show mls cef ip lookup 192.168.1.1 detail

Codes: M - mask entry, V - value entry, A - adjacency index, P - priority
bit
      D - full don't switch, m - load balancing modnumber, B - BGP Bucket
sel
      V0 - Vlan 0, C0 - don't comp bit 0, V1 - Vlan 1, C1 - don't comp bit 1
      RVTEN - RPF Vlan table enable, RVTSEL - RPF Vlan table select
Format: IPV4_DA - (8 | xtag vpn pi cr recirc tos prefix)
Format: IPV4_SA - (9 | xtag vpn pi cr recirc prefix)
M(85 ): E | 1 FFF 0 0 0 0 255.255.255.255
V(85 ): 8 | 1 0 0 0 0 0 192.168.1.1 (A:229376
,P:1,D:0,m:0 ,B:0 )
```

The `show mls cef adjacency entry <adjacency number>` command output shows hardware-level information related to the adjacency stored for a particular index number. In [Example 5-7](#), the index number is the value located after the "A:" notation. In [Example 5-8](#), you see the rewrite MAC addresses, the output VLAN ID, and the adjacency statistics for this entry.

Example 5-8. Step 6: Verifying the Hardware Adjacency Information

```
6500#show mls cef adjacency entry 229376 detail

Index: 229376 smac: 0013.5f1d.5000, dmac: 0000.0300.ffff
          mtu: 4488, vlan: 1021, dindex: 0x0, l3rw_vld: 1
          format: MAC_TCP, flags: 0x208408
          delta_seq: 0, delta_ack: 0
          packets: 0, bytes: 0
```

Example 5-8 shows the VLAN information listed as 1021. You can check which interface this maps to by looking at the show vlan internal usage data shown in **Example 5-9**. In this case, VLAN 1021 corresponds to Serial 3/0/0. The output interface should match the interface known through the routing table. This information matches in this case. If traffic was not passing and all this information is correct, confirm that the physical layer is working properly and determine that security access list or a special configuration is not blocking the traffic.

Example 5-9. Verifying Internal VLAN Usage Mapping

```
6500#show vlan internal usage

VLAN Usage
-----
1006 online diag vlan0
1007 online diag vlan1
1008 online diag vlan2
1009 online diag vlan3
1010 online diag vlan4
1011 online diag vlan5
1012 PM vlan process (trunk tagging)
1013 Control Plane Protection
1014 L3 multicast partial shortcuts for VPN 0
1015 Egress internal vlan
1016 Multicast VPN 0 QOS vlan
1017 IPv6 Multicast Egress multicast
1018 GigabitEthernet2/1
1019 GigabitEthernet2/2
1020 FastEthernet1/1
1021 Serial3/0/0
```

If a DFC-enabled card is the ingress interface, you must also look at the IOS FIB table on the line card. For example, traffic is ingress on module 4 and module 4 is DFC enabled, as shown in **Example 5-10**.

Example 5-10. Verifying That Module 4 Is Enabled with DFC

Code View: [Scroll](#) / [Show All](#)

```
6500-2#show module
Mod Ports Card Type Model Serial
No.
-----
-----
  3  48  CEF720 48 port 10/100/1000mb Ethernet WS-X6748-GE-TX
SAL083847JL
  4   4  CEF720 4 port 10-Gigabit Ethernet WS-X6704-10GE
SAL08528ADT
  5   2  Supervisor Engine 720 (Active) WS-SUP720-3BXL
SAD08320G9T
```

```

Mod MAC addresses                               Hw   Fw           Sw
Status
-----

```

```

 3 0011.bb78.7a94 to 0011.bb78.7ac3  2.1   12.2 (14r) S5  12.2 (18) SXF5  Ok
 4 0012.d944.2900 to 0012.d944.2903  2.0   12.2 (14r) S5  12.2 (18) SXF5  Ok
 5 0011.21b9.8bd4 to 0011.21b9.8bd7  4.0   8.1 (3)         12.2 (18) SXF5  Ok

```

```

Mod Sub-Module                               Model          Serial          Hw
Status
-----

```

```

 3 Centralized Forwarding Card WS-F6700-CFC      SAD081708B3    2.0    Ok
 4 Distributed Forwarding Card WS-F6700-DFC3A   SAD08260H6C    2.2    Ok
 5 Policy Feature Card 3       WS-F6K-PFC3BXL   SAD08310EJ5    1.3    Ok
 5 MSFC3 Daughterboard        WS-SUP720        SAD08300GD3    2.1    Ok

```

```

Mod Online Diag Status
-----

```

```

 3 Pass
 4 Pass
 5 Pass

```

You must use the remote login module command to view CEF entries or other statistical information on the DFC console. [Example 5-11](#) shows an example of verifying the software CEF and adjacency tables on a DFC-enabled module for prefix 10.9.8.66.

Example 5-11. Verifying Software CEF and Adjacency Tables on Module 4 Enabled with DFC

```

6500-2#remote command module 4 show ip cef 10.9.8.66 detail

10.9.8.66/32, version 34, epoch 0, connected, cached adjacency 10.9.8.66
0 packets, 0 bytes
  via 10.9.8.66, GigabitEthernet5/2, 0 dependencies
   next hop 10.9.8.66, GigabitEthernet5/2
   valid cached adjacency
6500-2#remote command module 4 show adjacency detail | begin 10.9.8.66
IP          GigabitEthernet5/2          10.9.8.66(5)
                                0 packets, 0 bytes
                                000BBF2FA4000008E24552400800
                                FIB LC      00:00:00
                                Epoch: 0

!output omitted for brevity

```

Hardware switching occurs with the hardware CEF entries but does not occur with the software CEF entries. [Example 5-12](#) shows checking hardware entries on the line card.

Example 5-12. Verifying the Hardware CEF and Adjacency Tables on Module 4 Enabled with DFC

Code View: [Scroll](#) / [Show All](#)

```
6500-2#show mls cef 10.9.8.66 detail module 4

Codes: M - mask entry, V - value entry, A - adjacency index, P - priority
bi
      D - full don't switch, m - load balancing modnumber, B - BGP Bucket
s
      V0 - Vlan 0, C0 - don't comp bit 0, V1 - Vlan 1, C1 - don't comp bit 1
      RVTEN - RPF Vlan table enable, RVTSEL - RPF Vlan table select
Format: IPV4_DA - (8 | xtag vpn pi cr recirc tos prefix)
Format: IPV4_SA - (9 | xtag vpn pi cr recirc prefix)
M(74      ): E | 1 FFF 0 0 0 0   255.255.255.255
V(74      ): 8 | 1 0   0 0 0 0   10.9.8.66           (A:163840 ,P:1,D:0,m:0
,
)
6500-2#show mls cef adjacency entry 163840 module 4

Index: 163840 smac: 0008.e245.5240, dmac: 000b.bf2f.a400
      mtu: 1518, vlan: 1029, dindex: 0x0, l3rw_vld: 1
      packets: 0, bytes: 0
```

The internal VLAN associated with the adjacency entry 163840 is VLAN 1029. [Example 5-13](#) shows that you can use the show vlan internal usage command to understand the mapping of the VLAN to the outbound interface.

Example 5-13. Verifying Outbound Interface

Code View: [Scroll](#) / [Show All](#)

```
6500-2#show vlan internal usage

VLAN Usage
-----
1006 online diag vlan0
1007 online diag vlan1
1008 online diag vlan2
1009 online diag vlan3
1010 online diag vlan4
1011 online diag vlan5
1012 PM vlan process (trunk tagging)
1013 Control Plane Protection
1014 L3 multicast partial shortcuts for VPN 0
1015 Egress internal vlan
1016 Multicast VPN 0 QOS vlan
1017 IPv6 Multicast Egress multicast
1018 GigabitEthernet5/1
1019 GigabitEthernet3/1
1020 GigabitEthernet3/5
1021 GigabitEthernet3/7
1022 GigabitEthernet3/8
1023 GigabitEthernet3/9
1024 GigabitEthernet3/13
1025 GigabitEthernet3/14
1026 GigabitEthernet3/15
```

```
1027 GigabitEthernet3/16
1029 GigabitEthernet5/2

6500-2#
```

In this case, the Multilayer Switching (MLS) hardware CEF entries match the software entries on the DFC and at the RP/SP level. The MAC destination address and output interface are also correct. From the troubleshooting shown in [Examples 5-11](#) through [Example 5-13](#), all is well at the hardware FIB level because the hardware CEF entries match the software adjacency and FIB table entries.

Troubleshooting Load Sharing

All features that are supported in software are not always supported in hardware. However, the software level supports load sharing and the hardware level also supports load sharing. Software limits the number of load-sharing paths supported per prefix in hardware. The number of load-sharing paths per prefix supported depends on the software code version. Refer to the Cisco IOS documentation for specific information for your version.

Per-packet load sharing is not supported on the SUP720. Hardware CEF load sharing is always per-destination. By default, hardware CEF performs per-destination load sharing using source and destination IP addresses. You can also configure per-destination load sharing to use Layer 4 information with the `mls ip cef load-sharing full` command.

By default, the SUP720 uses a unique system ID to prevent CEF polarization with load sharing, as discussed in [Chapter 6](#), "Load Sharing with CEF." To determine which next hop the SUP720 is using to forward a packet in hardware with multiple paths, use the `show mls cef exact-route` command to see the path. This command is the hardware equivalent of the `show ip cef exact-route` command. This can help you identify a physical issue with one path out of multiple paths for a particular destination.

[Table 5-1](#), earlier in this chapter, lists some helpful commands for troubleshooting connectivity issues on the SUP720.

[Example 5-14](#) shows how to look at hardware CEF statistics with the `show mls statistics` command. You can see information such as how many packets the SUP720 hardware switches.

Example 5-14. Verifying Hardware CEF Statistics

Code View: [Scroll](#) / [Show All](#)

```
6500-A#show mls statistics

Statistics for Earl in Module 6

L2 Forwarding Engine
  Total packets Switched           : 153772340

L3 Forwarding Engine
  Total packets L3 Switched        : 152381865 @ 125 pps

  Total Packets Bridged            : 432310
  Total Packets FIB Switched       : 431848
  Total Packets ACL Routed         : 0
  Total Packets Netflow Switched   : 0
  Total Mcast Packets Switched/Routed : 4258997
  Total ip packets with TOS changed : 2
  Total ip packets with COS changed : 2
  Total non ip packets COS changed : 0
  Total packets dropped by ACL     : 0
  Total packets dropped by Policing : 0
  Total packets exceeding CIR      : 0
```


Total packets exceeding PIR : 0

Errors

MAC/IP length inconsistencies : 0

Short IP packets received : 0

IP header checksum errors : 0

Total packets L3 Switched by all Modules: 152381865 @ 125 pps

Chapter 6. Load Sharing with CEF

This chapter includes the following topics:

- Benefits of load sharing
- Load sharing with process switching and fast switching
- Comparing CEF per-packet and CEF per-destination load sharing
- CEF architecture and load sharing
- CEF load sharing across parallel paths
- Per-packet load sharing on hardware-based platforms
- CEF per-packet load sharing on the Cisco GSR platform
- CEF load sharing troubleshooting examples

Understanding packet forwarding is vitally important for comprehension of load sharing across various Cisco platforms. When troubleshooting load sharing, it is easy to assume that there is a routing problem with Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), or Routing Information Protocol (RIP), for example, when it might just be a misunderstanding of what the expected results should be. Process switching, fast switching, Cisco Express Forwarding (CEF), and various types of hardware all handle load sharing differently. In addition, all can be viable options depending on the application.

Many people use the terms load sharing and load balancing to refer to the same option. However, with the term load balancing, many expect to have equal distributed loads across multiple paths. This book uses the term load sharing to describe more accurately what occurs with CEF and fast switching. In essence, load sharing is transmitting packets to a destination IP address over more than one path.

This chapter focuses on different types of load sharing with CEF and analyzes various configurable options. It also walks you through troubleshooting common problems on both software and hardware platforms.

Benefits of Load Sharing

Why would a company utilize load-sharing options across multiple links versus buying a bigger pipe? Well, for example, the customer might need more dedicated bandwidth than a T1 (1.5 Mbps), but a DS3 (45 Mbps) might be too much bandwidth and too costly for the customer's current needs. In other cases, a DS3 might not be available in the area. With multiple T1s, the customer can bond them together to utilize the aggregated bandwidth. Next, bonding and load sharing allow increased redundancy and reliability. The benefits of load sharing include link failover, lower costs, and more available bandwidth.

Network administrators can utilize methods such as CEF, multilink PPP (MLPPP), multilink frame relay (MFR), and inverse multiplexing over ATM (IMA) to bond T1s together. These methods allow customers to incrementally and cost-effectively grow bandwidth according to their company's needs. CEF has the capability to load-share through two options: per-packet and per-destination. MLPPP is an option to have bundled T1s behave like a single point-to-point link between an enterprise customer and an Internet service provider (ISP). IMA combines low-speed ATM circuits into a single logical pipe. [Table 6-1](#) discusses some of the characteristics of T1 bonding options. CEF is more efficient and causes less load on the CPU.

Table 6-1. Comparisons of T1 Bundling Options

Characteristic	Per-Packet CEF	Per-Destination CEF	Fast Switching	Per-Packet Process	Multilink PPP (MLPPP)
Out-of-order packets	Yes	No	No	Yes	No
Packets between source and destination follow same path	No	Yes	Yes	No	No
Router CPU load increase	No	No	No	Yes	Yes
Failover recovery	Yes	Yes	Yes	Yes	Yes

Load Sharing with Process Switching and Fast Switching

It is important to understand the difference between load sharing with process switching, fast switching, and CEF. Routers with newer Cisco IOS have CEF enabled by default. However, CEF must support the configured features. Process switching and fast switching both utilize the routing table to decide which path traffic will take to a destination. Therefore, the question is how to get the multiple paths in the routing table. For example, the routing table installs up to four equal-cost paths by default with dynamic routing protocols such as OSPF. You have the option to increase the number of equal-cost paths to a maximum value, depending on the particular Cisco IOS version and platform. Table 6-2 shows some of the options available.

Cisco IOS Releases	Maximum Paths Supported
Pre-12.0	6
12.0	
12.0T	
12.1	
12.1T	
12.2	
12.2T	
12.3	
12.0S	
12.0ST	

Table 6-2. Maximum Number of Paths Supported	
Cisco IOS Releases	Maximum Paths Supported
12.1E 12.2SX	
12.3T	16
12.2S	32

To use more than the default equal-cost paths with a dynamic routing protocol, you must use the maximum-path command. Under a particular routing protocol are configuration options for the maximum-path command. For example, to increase the number of equal cost paths that OSPF can install from four to six, use the maximum-path 6 command under the router ospf command. [Example 6-1](#) shows how to increase the maximum number of equal-cost paths installed in the routing table of Router R3 from four to six.

Example 6-1. Increasing the Maximum Equal-Cost Routes Allowed

```
R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#router ospf 100
R3(config-router)#maximum-paths ?
  <1-6>  Number of paths

R3(config-router)#maximum-paths 6
R3(config-router)#
```

[Example 6-2](#) shows that Router R3 is running Cisco IOS Release 12.2(5). Referring to [Table 6-2](#), R3 can install only up to six equal-cost paths with this Cisco IOS version. Installing multiple paths to a destination will increase the size of the routing table and CEF tables, and the amount of memory utilized. Therefore, use care when using this feature.

Example 6-2. Verifying the Cisco IOS Version

Code View: [Scroll](#) / [Show All](#)

```
R3#show version
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-JS-M), Version 12.2(5), RELEASE SOFTWARE
(fcl)
Copyright (c) 1986-2001 by Cisco Systems, Inc.
```

If it is utilizing static routing, the router will automatically install the maximum number of supported equal-cost routes without using the maximum-path command. If more than the maximum static routes are available, the routing table installs only the maximum number, as seen in [Example 6-3](#). Ten static routes are configured in Router R3. Based on the IOS version installed, Router R3 will only allow up to six paths, as seen earlier.

Therefore, Router R3's routing table only accepts the first six out of ten static routes to install, although the next hop is available for all ten of the static routes.

Note

EIGRP can also install unequal-cost paths in the routing table using the variance command.

Example 6-3. Only Six Static Routes Are Installed Although Ten Are Configured on Router R3

```
R3#show ip route static
      192.168.10.0/32 is subnetted, 5 subnets
S       192.168.10.10 [1/0] via 172.16.1.1
                   [1/0] via 172.16.1.2
                   [1/0] via 172.16.1.3
                   [1/0] via 172.16.1.4
                   [1/0] via 172.16.1.5
                   [1/0] via 172.16.1.6

R3#show run | begin ip route
ip route 192.168.10.10 255.255.255.255 172.16.1.1
ip route 192.168.10.10 255.255.255.255 172.16.1.2
ip route 192.168.10.10 255.255.255.255 172.16.1.3
ip route 192.168.10.10 255.255.255.255 172.16.1.4
ip route 192.168.10.10 255.255.255.255 172.16.1.5
ip route 192.168.10.10 255.255.255.255 172.16.1.6
ip route 192.168.10.10 255.255.255.255 172.16.1.7
ip route 192.168.10.10 255.255.255.255 172.16.1.8
ip route 192.168.10.10 255.255.255.255 172.16.1.9
ip route 192.168.10.10 255.255.255.255 172.16.1.10
!Output omitted for brevity.
```

Process switching will load-share across all equal-cost paths installed in the routing table. In the show ip route output, an asterisk represents the path that is currently in use for process switching. As seen in [Example 6-4](#), Router R1 has four paths to reach 10.1.6.1. The packet will take the path through Serial 0/3 if process switching occurs. If the asterisk does not move between paths after several captures of show ip route, this suggests that another switching mechanism is in operation. The asterisk does not mean that the router is only process switching. If a packet must be process switched instead of fast switched or CEF switched, the asterisk just shows the next hop to take in the process path. Selection of process switching on an interface results in load sharing on a per-packet basis, and the asterisk changes accordingly.

Example 6-4. show ip route Command Marks the Current Path Used by Process Switching with an Asterisk (*)

Code View: [Scroll](#) / [Show All](#)

```
R1#show ip route 10.1.6.1
Routing entry for 10.1.6.1/32
  Known via "eigrp 100", distance 90, metric 2300416, type internal
  Redistributing via eigrp 100
  Last update from 10.1.2.10 on Serial0/2, 4d20h ago
  Routing Descriptor Blocks:
  * 10.1.2.14, from 10.1.2.14, 4d20h ago, via Serial0/3
    Route metric is 2300416, traffic share count is 1
    Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
```

```

Loading 4/255, Hops 2
10.1.2.2, from 10.1.2.2, 4d20h ago, via Serial0/0
Route metric is 2300416, traffic share count is 1
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
Reliability 255/255, minimum MTU 1500 bytes
Loading 1/255, Hops 2
10.1.2.6, from 10.1.2.6, 4d20h ago, via Serial0/1
Route metric is 2300416, traffic share count is 1
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
Reliability 255/255, minimum MTU 1500 bytes
Loading 1/255, Hops 2
10.1.2.10, from 10.1.2.10, 4d20h ago, via Serial0/2
Route metric is 2300416, traffic share count is 1
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
Reliability 255/255, minimum MTU 1500 bytes
Loading 1/255, Hops 2

```

Although multiple paths for a destination can be installed in the routing table, fast switching uses only one path to build the cache entry. This means that the traffic will be load shared on a per-destination basis, not a per-packet basis. This limitation is because of the separation of routing and forwarding information with fast switching. [Example 6-5](#) shows a fast-cache entry created for the destination 10.1.6.1. In this case, fast switching uses only the path through interface Serial0/0 with next hop 10.1.2.6. Fast switching chooses a random path after the fast-cache path times out. This can be a different path or the same one. There is no way to do per-packet load sharing with fast switching. Therefore, with fast switching, uneven link utilization is possible if most of the traffic goes to the same destination, even though multiple paths exist.

Example 6-5. Fast-Switching Entry Created Using One Path Only with Multiple Routing Paths

```

R1#show ip cache
IP routing cache 2 entries, 340 bytes
  4 adds, 2 invalidates, 0 refcounts
Minimum invalidation interval 2 seconds, maximum interval 5 seconds,
quiet interval 3 seconds, threshold 0 requests
Invalidation rate 0 in last second, 0 in last 3 seconds
Last full cache invalidation occurred 4d21h ago

Prefix/Length      Age           Interface     Next Hop
10.1.6.1/32        00:01:15    Serial0/1     10.1.2.6

```

Comparing CEF Per-Packet and CEF Per-Destination Load Sharing

CEF uses the routing table to build its Forwarding Information Base (FIB) table. If multiple paths to a destination exist in the routing table, there will be multiple paths to a destination in the FIB table. CEF does not care which routing protocol placed the routes in the routing table. As with process switching and fast switching, the same rules apply to CEF to get multiple paths in the routing table.

Understanding Per-Destination Load Sharing

By default, CEF does per-destination load sharing. CEF per-destination load sharing is not the same as fast switching per-destination load sharing. This chapter discusses CEF operation prior to Cisco IOS Release 12.2S. Changes occurred in CEF operation with Cisco IOS 12.2S. [Chapter 3](#), "CEF Enhanced Scalability," discusses these changes.

With CEF, the router maps a set of destination paths to 16 buckets. Depending on the number of paths to a destination, the router might or might not assign all the buckets. A hash function determines which path a particular source-destination pair will take.

CEF per-destination load sharing assigns all packets for a specific source-destination pair to one of the paths. Traffic between this particular source-destination pair will always take the same path. So essentially, CEF per-destination load sharing is really per-session or per-flow load sharing. The multiple paths for a particular destination will share the load based on the CEF hash determination.

The advantage of CEF per-destination load sharing is that packets for each session (source-destination pair) maintain their order and take the same path to the destination. This avoids some of the application problems that can be seen with CEF per-packet load sharing. However, with CEF per-destination load sharing, the router could overload one link if one of the sessions is sending a huge amount of traffic. Again, CEF bases per-destination load sharing on source-destination pairs. All packets exiting the router with the same source-destination pair will always take the same path. If Network Address Translation (NAT) is taking place elsewhere in the network so that all packets appear to come from the same source and go to the same destination, traffic will always take the same path. Even with a large number of source-destination pairs, uneven distribution can also exist if there are large flows that take one path and other flows that take another path. However, with a large number of source-destination pairs and even number of flows, CEF distributes the load well over equal paths. CEF per-destination load sharing works well in most scenarios.

Understanding Per-Packet Load Sharing

CEF per-packet load sharing is a configurable option for Cisco routers. Per-packet load sharing sends successive packets in a round-robin fashion without regard to the packet source or destination. Many ISPs are willing to run CEF per-packet load sharing with directly connected customers to provide aggregation. This allows a customer to grow the network gradually rather than to have to upgrade immediately to a DS3 before the demand for such a bandwidth occurs. Per-packet load sharing also helps to ensure that one path does not get overloaded.

Path utilization with CEF per-packet load sharing can be effective. However, total link usage can vary depending on how consistent the packet sizes are. Per-packet load sharing sends packets in a round-robin fashion without checking the packet size. So, a flow of big packets as compared to a flow of small packets can make a big difference on path utilization. Also, for a given source-destination pair, traffic can take different paths and lead to issues with out-of-order packets and latency. These issues can decrease the effectiveness of per-packet load sharing by causing TCP retransmissions, TCP throttles, and application timeouts. Many users essentially see poor throughput with per-packet load sharing because of out-of-order packets. Also, per-packet load sharing is not available on all platforms. Check with your Cisco account representative or [Cisco.com](https://www.cisco.com) for the latest feature support information.

Minimizing Out-of-Order Packets

If a customer has one router with several T1s connected to the same ISP's router, per-packet load sharing can be a viable solution. It is important that the T1s all start on same router and end on the same ISP router to try to eliminate out-of-order packets. Trying to do per-packet load sharing across multiple T1s going across different ISPs is definitely asking for out-of-order packets.

Another consideration with running per-packet load sharing is to make sure that the circuits are the same, that is, the same speed, same ISP, and so on. With dissimilar circuits, there is a greater chance of having out-of-order packets.

With sensitive applications such as voice, CEF per-destination load sharing is a better choice than CEF per-packet load sharing. While the network might show good traffic distribution with CEF per-packet load sharing, the actual useful throughput can be less than the throughput with a single link.

Configuring CEF Per-Packet Load Sharing

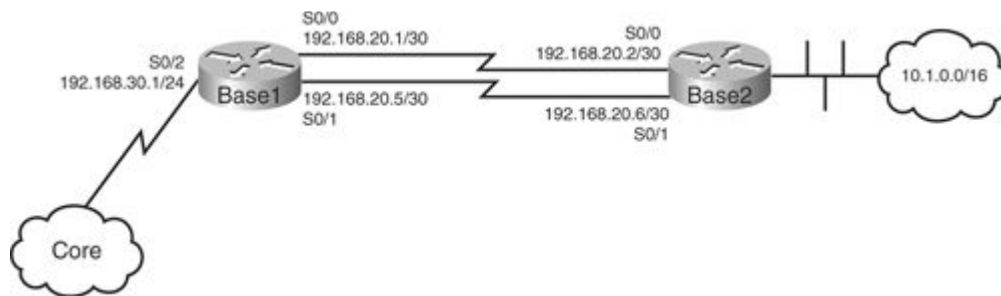
A Cisco router chooses whether to CEF switch based on whether the inbound interface has CEF enabled. However, the router makes load-sharing decisions based on the outbound interface. By default, the outbound interface's configuration is per-destination. Therefore, for per-packet load sharing, the outbound interface must contain the per-packet load-sharing configuration on most Cisco routers. Certain line cards can be different on

the Cisco Gigabit Switch Router (GSR) platform. Refer to the documentation on Cisco.com for information about configuring per-packet load sharing on Cisco GSR platforms.

Figure 6-1 shows two equal-cost paths for destination network 10.1.0/16. In Example 6-6, notice that Router Base1's configurations include enabling per-packet load sharing on both outbound interfaces Serial 0/0 and Serial 0/1.

Figure 6-1. Simple CEF Per-Packet Load-Sharing Network

[\[View full size image\]](#)



Example 6-6. Configuring CEF Per-Packet Load Sharing on Router Base1

```
Base1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Base1(config)#interface serial0/0
Base1(config-if)#ip load-sharing per-packet
Base1(config-if)#exit
Base1(config)#
Base1(config)#interface serial0/1
Base1(config-if)#ip load-sharing per-packet
```

Check the CEF per-packet configuration with the `show cef interface interface` command. Notice that per-packet load sharing is in the enabled state for Router Base1 in Example 6-7.

Example 6-7. Verifying CEF Per-Packet Configuration on Base1's Serial 0/0 and Serial 0/1 Interfaces

```
Base1#show cef interface serial 0/0
Serial0/0 is up (if_number 4)
  Internet address is 10.1.2.1/30
  ICMP redirects are always sent
  Per packet load-sharing is enabled
!Output omitted for brevity
Base1#
Base1#show cef interface serial 0/1
Serial0/1 is up (if_number 5)
  Internet address is 10.1.2.5/30
  ICMP redirects are always sent
  Per packet load-sharing is enabled
!Output omitted for brevity
```

You can also check per-packet load sharing with the `show ip cef <prefix>` command, as seen in Example 6-8.

Example 6-8. Verifying Per-Packet Load Sharing Through the Destination Prefix

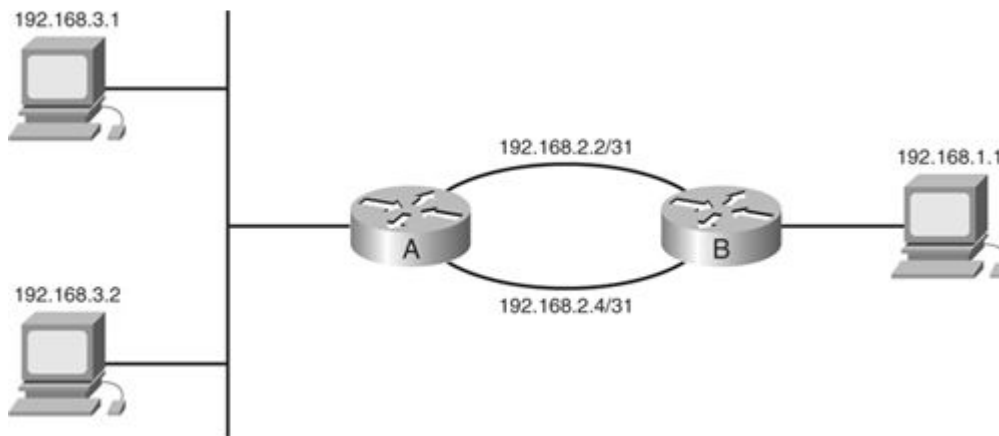
```
Basel#show ip cef 10.1.0.0
10.1.0.0/16, version 176, per-packet sharing
0 packets, 0 bytes
  via 192.168.20.2 Serial0/0, 0 dependencies
    traffic share 1, current path
    next hop 192.168.20.2, Serial0/0
    valid adjacency
  via 192.168.20.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 192.168.20.6, Serial0/1
    valid adjacency
0 packets, 0 bytes switched through the prefix
```

Reverting to CEF per-destination load sharing is easy. On the outbound interfaces, configure the ip load-share per-destination command.

CEF Architecture and Load Sharing

In [Figure 6-2](#), Router A has two paths to switch packets that 192.168.3.1 and 192.168.3.2 send toward 192.168.1.1. The router installs both of these paths in the routing table.

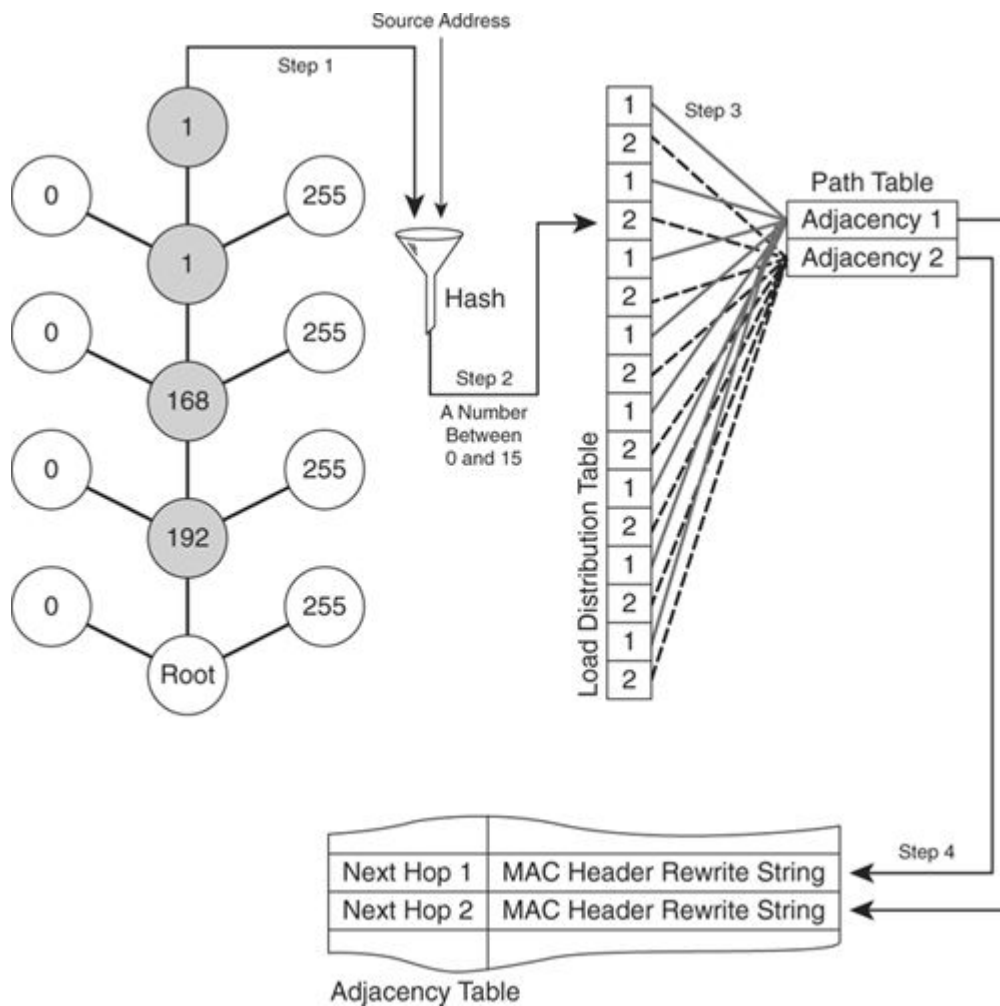
Figure 6-2. Simple Network with Multiple Paths
[\[View full size image\]](#)



How does CEF know that there are multiple paths for 192.168.1.1? CEF knows this from the path information it receives from the routing table. One other item that CEF receives from the routing table is the traffic-share allocation. The traffic-share value weighs certain paths more than other paths. This is of particular importance to unequal load sharing. The router maps paths to 16 buckets to determine which path the traffic will follow. There are a total of 16 buckets in CEF. The router fills the 16 buckets with active paths based on the traffic-share value. CEF places a different number of path entries in the load distribution table based on its weight (traffic-share value). According to the load distribution table, CEF assigns more sessions to the heavier-weighted paths to use them more than other paths in the instance of unequal route installation.

In [Figure 6-3](#), a hash determines which bucket to use to choose the final path for a destination. In this case, the load distribution table uses the two paths a total of eight times each because the paths have equal weights.

Figure 6-3. CEF Load-Sharing Data Structures
[\[View full size image\]](#)



The following list explains the process illustrated in [Figure 6-3](#):

1. As a packet enters the inbound interface, the router looks up the destination address in CEF. The load distribution table indicates that there are multiple paths to this destination. In this case, two equal-cost paths exist.
2. CEF combines and hashes the source and destination addresses along with a unique ID (depending on the Cisco IOS version) to find a number between 0 and 15. This number is an index into the load distribution table.
3. The load distribution table then indicates which adjacency entry in the path table should be used based on the hash information.
4. The path table then points into the adjacency table, so the router uses the correct next hop to reach the destination.

CEF Load Sharing Across Parallel Paths

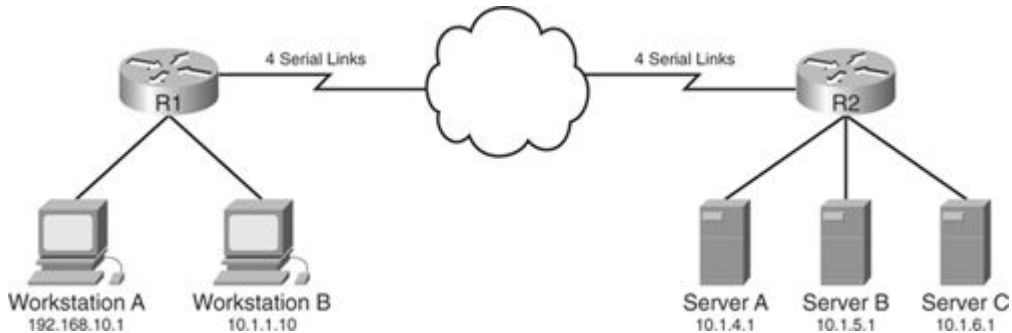
Companies commonly have parallel paths to remote destinations. CEF can allow load sharing across both paths instead of only using one path for packet transport. Which method a network administrator uses depends on the applications traversing the parallel links and their tolerances.

CEF Per-Destination Example

This section describes how CEF handles per-destination load sharing across parallel paths. Figure 6-4 shows four parallel paths between Routers R1 and R2. So, Router R1 has four paths to switch packets that 192.168.10.1 sends toward 10.1.6.1.

Figure 6-4. Sample Network for Four Parallel Paths

[\[View full size image\]](#)



With CEF enabled, per-destination load sharing will occur by default. In the shaded portion of Example 6-9, the `show ip cef <destination prefix>` command shows that the sharing state is per-destination sharing.

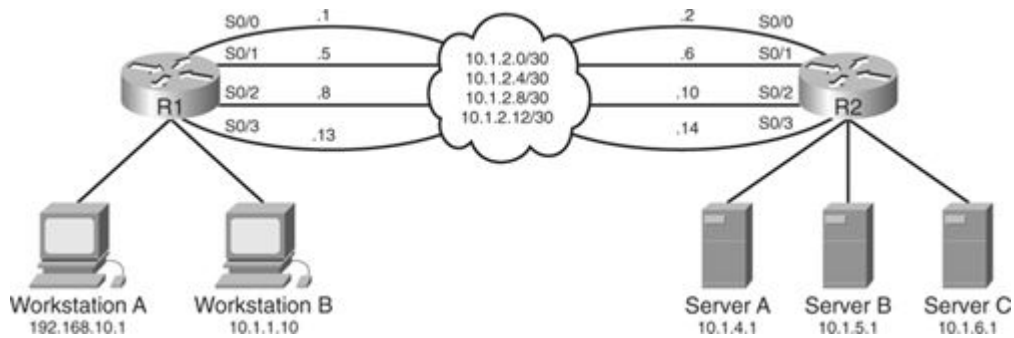
Example 6-9. Verifying CEF Per-Destination Load Sharing

Code View: [Scroll](#) / [Show All](#)

```
R1#show ip cef 10.1.6.1
10.1.6.1/32, version 42, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 1
    next hop 10.1.2.2, Serial0/0
    valid adjacency
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 10.1.2.6, Serial0/1
    valid adjacency
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
R1#
```

Figure 6-5 shows the physical configuration between Routers R1 and R2.

Figure 6-5. Detailed Network for Load Sharing Across Parallel Paths
[\[View full size image\]](#)



As seen in [Example 6-10](#), the traffic-share value for each path to 192.168.10.4 is 1 because this is equal-cost load sharing. The traffic-share values build the load distribution table also shown in [Example 6-10](#). CEF uses the load distribution table to distribute traffic to a particular destination across the multiple paths. In [Example 6-10](#), each of the four paths is in the load distribution table four times.

Note

The show ip cef <prefix> internal command is a hidden command in many Cisco IOS versions.

Example 6-10. Verifying CEF Load Distribution Table

Code View: [Scroll](#) / [Show All](#)

```
R1#show ip cef 10.1.6.1 internal
10.1.6.1/32, version 42, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 1
    next hop 10.1.2.2, Serial0/0
    valid adjacency
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 10.1.2.6, Serial0/1
    valid adjacency

0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
Load distribution: 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 (refcount 1)

Hash  OK  Interface          Address           Packets
1     Y   Serial0/2         point2point      0
```

2	Y	Serial0/3	point2point	0
3	Y	Serial0/0	point2point	0
4	Y	Serial0/1	point2point	0
5	Y	Serial0/2	point2point	0
6	Y	Serial0/3	point2point	0
7	Y	Serial0/0	point2point	0
8	Y	Serial0/1	point2point	0
9	Y	Serial0/2	point2point	0
10	Y	Serial0/3	point2point	0
11	Y	Serial0/0	point2point	0
12	Y	Serial0/1	point2point	0
13	Y	Serial0/2	point2point	0
14	Y	Serial0/3	point2point	0
15	Y	Serial0/0	point2point	0
16	Y	Serial0/1	point2point	0

R1#

So how do you know which path a particular session will take? The `show ip cef exact-route <source address> <destination address>` command shows this information. For example, if an FTP xxx is occurring from 192.168.10.1 to 10.1.6.1 and there is an FTP xxx from 10.1.1.10 to 10.1.6.1, will both traffic flows take the same path with CEF per-destination load sharing? It is unlikely, but it is possible. Depending on the number of sessions and paths, each session will most likely take different paths. With fast switching, both traffic flows will take the same path because the same destination is the target. From [Example 6-11](#), traffic from 192.168.10.1 to 10.1.6.1 will take the path through Serial 0/3. The FTP traffic session from 10.1.1.10 to 10.1.6.1 will take the path through Serial 0/0.

Example 6-11. Verifying CEF Path for a Session

```
R1#show ip cef exact-route 192.168.10.1 10.1.6.1
192.168.10.1    -> 10.1.6.1      : Serial0/3 (next hop 10.1.2.14)
R1#show ip cef exact-route 10.1.1.10 10.1.6.1
10.1.1.10     -> 10.1.6.1      : Serial0/0 (next hop 10.1.2.2)
```

Even if a user starts a Telnet session from 10.1.1.10 to 10.1.6.1, traffic would still flow through Serial 0/0. CEF does not discriminate based on Layer 4 port information. CEF only looks at source-destination pairs to determine whether the flow is a session and which path it should take. Also, if traffic is heavy from 10.1.1.10 to 10.1.6.1, Serial 0/0 could be overloaded while the other serial links are light.

Consider load distribution to server B with IP address 10.1.5.1 in [Figure 6-5](#) from various sources ranging from 10.1.1.10 to 10.1.1.135. The best way to examine whether load sharing is occurring is to compare the transmit load (txload) in the `show interfaces` command. In [Example 6-12](#), the txload for Serial 0/0, Serial 0/1, Serial 0/2, and Serial 0/3 is the same at 3/255.

Not only is the txload the same, but the packet output rate from the serial interfaces is also relatively the same. This is also an indication of traffic distribution.

Example 6-12. Verifying CEF Per-Destination Load Distribution

Code View: [Scroll](#) / [Show All](#)

```
R1#show interfaces
!Output omitted for brevity
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
```

```
Internet address is 10.1.2.1/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:02, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:22:02
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 21000 bits/sec, 29 packets/sec
 1598 packets input, 98442 bytes, 0 no buffer
  Received 154 broadcasts, 0 runts, 0 giants, 0 throttles
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
 40016 packets output, 3589874 bytes, 0 underruns
  0 output errors, 0 collisions, 0 interface resets
  0 output buffer failures, 0 output buffers swapped out
  0 carrier transitions
 DCD=up DSR=up DTR=up RTS=up CTS=up
```

```
Serial0/1 is up, line protocol is up
Hardware is PowerQUICC Serial
Internet address is 10.1.2.5/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:22:03
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
30 second input rate 0 bits/sec, 1 packets/sec
30 second output rate 21000 bits/sec, 30 packets/sec
 1156 packets input, 72053 bytes, 0 no buffer
  Received 154 broadcasts, 0 runts, 0 giants, 0 throttles
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
 40053 packets output, 3593624 bytes, 0 underruns
  0 output errors, 0 collisions, 0 interface resets
  0 output buffer failures, 0 output buffers swapped out
  0 carrier transitions
 DCD=up DSR=up DTR=up RTS=up CTS=up
```

```
Serial0/2 is up, line protocol is up
Hardware is PowerQUICC Serial
Internet address is 10.1.2.9/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:22:04
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 24000 bits/sec, 34 packets/sec
  793 packets input, 50137 bytes, 0 no buffer
  Received 154 broadcasts, 0 runts, 0 giants, 0 throttles
```

```

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
44866 packets output, 4026460 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions
DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/3 is up, line protocol is up
Hardware is PowerQUICC Serial
Internet address is 10.1.2.13/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:02, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:22:05
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 23000 bits/sec, 32 packets/sec
881 packets input, 55265 bytes, 0 no buffer
Received 155 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
42011 packets output, 3769726 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions
DCD=up DSR=up DTR=up RTS=up CTS=up

```

The load interval for each interface is set to 30 seconds to accurately represent the load. The default value for the load interval is 5 minutes for each interface. If the load interval is set to 30 seconds, the router computes the average statistics for the last 30 seconds of load data instead of the entire 5 minutes. The input/output statistics from the show interface command will be more current with a lower load interval. Changing the load interval will have minimal impact on the router's performance. To change the load interval, use the following interface command:

```

R1(config-if)#load-interval ?
<30-600> Load interval delay in seconds

```

Another way to look at the traffic distribution is to enable the global, hidden command `ip cef accounting load-balance-hash`. Enabling CEF load accounting can help to check the number of packets the router sends on each path. This global command is good for short periods to gather information concerning the load distribution. Wait until some traffic passes after enabling CEF load accounting and check the packets that went into each bucket with the hidden `show ip cef <destination prefix> internal` command. Now, looking at the `show ip cef <destination prefix> internal` output, you will see the packets sent through each bucket and interface. In [Example 6-13](#), there are 16 hash buckets for the destination 10.1.5.1. The four serial interfaces on the router each map to four different hash buckets.

Example 6-13. Verifying CEF Per-Destination Distribution Through Accounting

Code View: [Scroll](#) / [Show All](#)

```

R1#show ip cef 10.1.5.1 internal
10.1.5.1/32, version 56, epoch 0, per-destination sharing

```

```

0 packets, 0 bytes
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 10.1.2.6, Serial0/1
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 1
    next hop 10.1.2.2, Serial0/0
    valid adjacency
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency

0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
Load distribution: 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 (refcount 1)

Hash  OK  Interface                Address                Packets
 1     Y   Serial0/3                point2point            1672
 2     Y   Serial0/1                point2point            1115
 3     Y   Serial0/0                point2point            2511
 4     Y   Serial0/2                point2point             557
 5     Y   Serial0/3                point2point            1672
 6     Y   Serial0/1                point2point            3343
 7     Y   Serial0/0                point2point            1670
 8     Y   Serial0/2                point2point            2787
 9     Y   Serial0/3                point2point            3897
10     Y   Serial0/1                point2point            3345
11     Y   Serial0/0                point2point            1113
12     Y   Serial0/2                point2point            3904
13     Y   Serial0/3                point2point            1674
14     Y   Serial0/1                point2point             557
15     Y   Serial0/0                point2point            2798
16     Y   Serial0/2                point2point            2240
R1#

```

To examine the distribution, you can add all the packets that went through a particular interface. For example, for interface Serial 0/3, when adding the packets sent through hash buckets 1, 5, 9, and 13, a sum of 8915 packets are going through Serial 0/3. Table 6-3 shows the total number of packets sent through each interface in this CEF per-destination example.

Output Interface	Hash Buckets	Total Packets Sent
Serial 0/0	3, 7, 11, 15	8092

Output Interface	Hash Buckets	Total Packets Sent
Serial 0/1	2, 6, 10, 14	8360
Serial 0/2	4, 8, 12, 16	9438
Serial 0/3	1, 5, 9, 13	8915

CEF Per-Packet Example

Now, consider CEF per-packet load sharing on the same network with the same flows. (Refer to [Figure 6-5](#).) In [Example 6-14](#), the txload is 3/255 for each of the output interfaces. The packets/second rate is also 31 for each of the output interfaces. Remember that although the distribution can be equal with per-packet load sharing regardless of the source or destination, other problems can occur at the application layer that affect performance.

Example 6-14. Verifying CEF Per-Packet Load Distribution

Code View: [Scroll](#) / [Show All](#)

```
R1#show interfaces
!Output omitted for brevity
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 10.1.2.1/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:14:20
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
    Conversations 0/4/256 (active/max active/max total)
    Reserved Conversations 0/0 (allocated/max allocated)
    Available Bandwidth 1158 kilobits/sec
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 22000 bits/sec, 31 packets/sec
  742 packets input, 46343 bytes, 0 no buffer
  Received 100 broadcasts, 0 runts, 0 giants, 0 throttles
  0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
  25573 packets output, 2294378 bytes, 0 underruns
  0 output errors, 0 collisions, 1 interface resets
  0 output buffer failures, 0 output buffers swapped out
  0 carrier transitions
  DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 10.1.2.5/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
    reliability 255/255, txload 3/255, rxload 1/255
```

```
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:03, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:14:21
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
  Conversations 0/9/256 (active/max active/max total)
  Reserved Conversations 0/0 (allocated/max allocated)
  Available Bandwidth 1158 kilobits/sec
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 22000 bits/sec, 31 packets/sec
691 packets input, 43152 bytes, 0 no buffer
Received 100 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
24011 packets output, 2153352 bytes, 0 underruns
0 output errors, 0 collisions, 1 interface resets
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions
DCD=up DSR=up DTR=up RTS=up CTS=up
```

```
Serial0/2 is up, line protocol is up
Hardware is PowerQUICC Serial
Internet address is 10.1.2.9/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
  reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:14:23
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
  Conversations 0/5/256 (active/max active/max total)
  Reserved Conversations 0/0 (allocated/max allocated)
  Available Bandwidth 1158 kilobits/sec
30 second input rate 0 bits/sec, 1 packets/sec
30 second output rate 22000 bits/sec, 31 packets/sec
697 packets input, 43507 bytes, 0 no buffer
Received 100 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
24850 packets output, 2229084 bytes, 0 underruns
0 output errors, 0 collisions, 1 interface resets
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions
DCD=up DSR=up DTR=up RTS=up CTS=up
```

```
Serial0/3 is up, line protocol is up
Hardware is PowerQUICC Serial
Internet address is 10.1.2.13/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
  reliability 255/255, txload 3/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters 00:14:24
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: weighted fair
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
  Conversations 0/2/256 (active/max active/max total)
```

```

Reserved Conversations 0/0 (allocated/max allocated)
Available Bandwidth 1158 kilobits/sec
30 second input rate 0 bits/sec, 0 packets/sec
30 second output rate 22000 bits/sec, 31 packets/sec
599 packets input, 37533 bytes, 0 no buffer
Received 101 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
23656 packets output, 2121618 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions
DCD=up DSR=up DTR=up RTS=up CTS=up

```

Again, another way to look at the traffic distribution is to enable the global, hidden command `ip cef accounting load-balance-hash`. Enabling CEF load accounting can help to check the number of packets the router sends on each path for CEF per-packet load sharing. [Example 6-15](#) shows the output of the `show ip cef <destination prefix> internal` command.

Example 6-15. Verifying CEF Per-Packet Distribution Through Accounting

Code View: [Scroll](#) / [Show All](#)

```

R1#show ip cef 10.1.5.1 internal
10.1.5.1/32, version 56, epoch 0, per-packet sharing
0 packets, 0 bytes
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 10.1.2.6, Serial0/1
    valid adjacency
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 1, current path
    next hop 10.1.2.2, Serial0/0
    valid adjacency

0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
Load distribution: 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 (refcount 1)

Hash  OK  Interface          Address           Packets
 1     Y   Serial0/1         point2point      2071
 2     Y   Serial0/2         point2point      2071
 3     Y   Serial0/3         point2point      2071
 4     Y   Serial0/0         point2point      2071
 5     Y   Serial0/1         point2point      2071
 6     Y   Serial0/2         point2point      2071
 7     Y   Serial0/3         point2point      2071
 8     Y   Serial0/0         point2point      2071

```

9	Y	Serial0/1	point2point	2071
10	Y	Serial0/2	point2point	2071
11	Y	Serial0/3	point2point	2071
12	Y	Serial0/0	point2point	2071
13	Y	Serial0/1	point2point	2071
14	Y	Serial0/2	point2point	2071
15	Y	Serial0/3	point2point	2071
16	Y	Serial0/0	point2point	2071
R1#				

To examine the distribution, you can add all the packets that went through a particular interface. For example, for interface Serial 0/1, when adding the packets sent through hash buckets 1, 5, 9, and 13, a sum of 8284 packets are going through Serial 0/3. Table 6-4 shows the total number of packets sent through each interface in this CEF per-packet example.

Table 6-4. Total Packets Sent Through Interfaces in Example 6-15

Output Interface	Hash Buckets	Total Packets Sent
Serial 0/0	4, 8, 12, 16	8284
Serial 0/1	1, 5, 9, 13	8284
Serial 0/2	2, 6, 10, 14	8284
Serial 0/3	3, 7, 11, 15	8284

Per-Packet Load Sharing on Hardware-Based Platforms

A big performance benefit of Cisco Catalyst switches is that they do most of the switching in hardware and not in software. Therefore, the key is to remember that the software-switching parameters are not applicable to the hardware-switched path. For example, the commands for configuring the ip load-sharing per-packet command on the Multilayer Switch Feature Card 2 (MSFC2) for the Catalyst 6500/7600 platforms only applies to traffic that is CEF switched on the MSFC2. This command does not affect traffic that is hardware switched on the Policy Feature Card 2 (PFC2) or Distributed Forwarding Card (DFC) modules. Hardware-based platforms such as Catalyst 3550 series switches and Catalyst 6500/7600 series switches do not support CEF per-packet load sharing. Hardware-based CEF uses per-flow load sharing.

On a Catalyst 6500/7600 MSFC2 or a Supervisor 720 (SUP720), you can use Layer 4 information for load sharing by enabling the mls ip cef load-sharing full command in native mode or the set mls load-balance full command in hybrid mode. Load sharing based on Layer 4 TCP/UDP ports in addition to source-destination can offer some granularity. Note that the software maximum supported paths also limit the hardware maximum supported paths for a destination. For example, on a Catalyst 6500 running Cisco IOS Release 12.1E, multilayer switching on a PFC2 will only support eight equal-cost paths because of the limitation of the Cisco IOS version, although the hardware is capable of 16 adjacencies.

Originally, only platforms that used software-based forwarding supported CEF per-packet load sharing. Now some platforms that use Parallel Express Forwarding (PXF) honor CEF per-packet load sharing and can do this in hardware. Some PXF-based platforms, such as the Cisco 10000 ESR, support per-packet load sharing.

However, a Cisco 7300 series router with an NSE-100 does not support CEF per-packet load sharing. Refer to Cisco.com for the latest information regarding feature support.

CEF Per-Packet Load Sharing on the Cisco GSR Platform

A Cisco Gigabit Switch Router (GSR) platform enables CEF by default and for proper forwarding operation. A Cisco GSR's per-packet load-sharing capabilities depend on the ingress line cards installed and the configuration. Table 6-5 highlights the main differences in the engines and their per-packet capabilities.

Table 6-5. Per-Packet Load Sharing for GSR Engine Types	
Engine Type	Per-Packet Load-Sharing Characteristics
0	With proper egress configuration, per-packet switching occurs for packets inbound on this line card.
1	With proper egress configuration, per-packet switching occurs for packets inbound on this line card.
2	Per-ingress line card configuration required. With proper Engine 2 line card configuration, per-packet switching occurs regardless of egress configuration for packets inbound on this line card. Not all Engine 2 cards support this.
3	Inbound packets will not be per-packet switched. If the ingress card such as an Engine 2 per-packet switches to this card, per-packet load sharing can occur for exiting packets.
4	Inbound packets will not be per-packet switched. If the ingress card such as an Engine 2 per-packet switches to this card, per-packet load sharing can occur for exiting packets.
4+	Per-ingress line card configuration required. With proper Engine 4+ line card configuration, per-packet switching occurs regardless of egress configuration for packets inbound on this line card.
6	Per-ingress line card configuration required. With proper Engine 6 line card configuration, per-packet switching occurs regardless of egress configuration for packets inbound on this line card.

In general, on the Cisco GSR series, per-packet load sharing is available on all Layer 3 forwarding engines except Engines 3 and 4.

For packets entering on an Engine 0 or Engine 1 line card, the per-packet load sharing will occur if the multiple egress interfaces have the proper configuration. As long as the equal-cost destination interfaces have the ip load-sharing per-packet command under the interface, load sharing will occur on a round-robin basis.

For packets entering on an Engine 2 or Engine 4+ card, different rules apply. In this case, each ingress line card makes independent forwarding decisions and must have configuration for per-packet load sharing. Therefore, the ingress must support per-packet load sharing for sharing to work properly. If the Engine 2 or Engine 4+ line cards are set to per-packet mode, forwarding will occur for all packets entering the line card to the egress interfaces in a round-robin fashion, regardless of the configuration on the egress interface. So, if the Engine 2 or Engine 4+ card is set to per-packet load sharing, packets will be per-packet switched regardless of whether the egress has ip load-sharing per-packet configured.

Per-packet load-sharing configuration on an Engine 2 or Engine 4+ card is different from other routers. For an Engine 2 or Engine 4+ line card, you must use the `hw-module slot <number> ip load-sharing per-packet` command to configure per-packet load sharing. This command has no effect on other line card engine types. There is one other rule for Engine 2 line cards. Only POS Engine 2 line cards support per-packet load sharing. There is no per-packet load-sharing support on Engine 2 line cards such as the 3-port Gigabit Ethernet Card. Per-packet load sharing will not occur for packets entering on an Engine 3 or an Engine 4 line card. Per-packet load sharing can occur for packets exiting these line cards if the ingress per-packet switches to these interfaces.

To view the exact route for each IP flow, use the `exec slot <slot number> show ip hardware-CEF exact-route <src> <dst>` command on line cards that use hardware-based CEF tables.

CEF Load-Sharing Troubleshooting Examples

Many concerns with CEF load sharing occur because there is a lack of understanding of how it operates. This section describes some common issues seen when troubleshooting CEF load-sharing problems or perceived problems.

CEF Per-Destination Load Sharing Overloading One Link

In certain cases, CEF per-destination load sharing will overload one link when multiple paths to a destination exist. To understand why this happens, it is important to understand CEF operation. When CEF chooses a path, it does not take the volume of traffic into account, so a heavy flow between one source and destination pair can easily overload a link.

Sometimes a particular design is not suitable for CEF per-destination load sharing. If traffic primarily goes from one source to one destination, the traffic will always take the same path. Again, refer to Figure 6-5 in the section "CEF Per-Destination Example," earlier in this chapter.

Assume that user 10.1.1.10 sends a large amount of traffic to server 10.1.5.1. The router will only use one path with CEF per-destination enabled. In this case, the session from 10.1.1.10 to 10.1.5.1 uses Serial 0/1, as seen with the `show ip cef exact-route` output:

```
R1#show ip cef exact-route 10.1.1.10 10.1.5.1  
  
10.1.1.10    -> 10.1.5.1    : Serial0/1 (next hop 10.1.2.6)
```

In Example 6-16, you can see that Serial 0/1 has a higher load than the rest of the interfaces. Serial 0/0 and Serial 0/2 do not have any load. In a scenario utilizing Network Address Translation (NAT), traffic can appear to come from one source and go to the same destination. It would be easy to overutilize one link, but the other equal-cost paths would not be utilized. This would be an inefficient use of resources and a waste of money if you want to use all links at all times. In this type of layout, you might need to investigate using options such as policy-based routing or multilink PPP (MLPPP), or only have one bigger output link.

Example 6-16. Uneven Distribution with One Primary User

```
Code View: Scroll / Show All  
  
R1#show interfaces  
  
!Output omitted for brevity  
  
Serial0/0 is up, line protocol is up
```

Hardware is PowerQUICC Serial

Internet address is 10.1.2.1/30

MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
reliability 255/255, txload 0/255, rxload 1/255

!Output omitted for brevity

30 second input rate 1000 bits/sec, 2 packets/sec

30 second output rate 0 bits/sec, 0 packets/sec

707 packets input, 43120 bytes, 0 no buffer

Received 37 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

103 packets output, 6542 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/1 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.5/30

MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
reliability 255/255, txload 4/255, rxload 1/255

!Output omitted for brevity

30 second input rate 0 bits/sec, 0 packets/sec

30 second output rate 25000 bits/sec, 35 packets/sec

105 packets input, 7004 bytes, 0 no buffer

Received 37 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

11004 packets output, 987606 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/2 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.9/30

MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,

reliability 255/255, txload 0/255, rxload 1/255

!Output omitted for brevity

30 second input rate 0 bits/sec, 0 packets/sec

30 second output rate 0 bits/sec, 0 packets/sec

105 packets input, 7004 bytes, 0 no buffer

Received 37 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

104 packets output, 6566 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/3 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.13/30


```
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,  
  reliability 255/255, txload 1/255, rxload 1/255  
!Output omitted for brevity  
  
30 second input rate 0 bits/sec, 0 packets/sec  
30 second output rate 0 bits/sec, 1 packets/sec  
  105 packets input, 7004 bytes, 0 no buffer  
Received 38 broadcasts, 0 runts, 0 giants, 0 throttles  
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort  
420 packets output, 34980 bytes, 0 underruns  
0 output errors, 0 collisions, 0 interface resets  
0 output buffer failures, 0 output buffers swapped out  
0 carrier transitions  
DCD=up DSR=up DTR=up RTS=up CTS=up
```

Looking at Example 6-17 for destination prefix 10.1.5.1, there are four paths known with CEF per-destination load sharing operating. However, most of the load is through Serial 0/1 and hash bucket 3. The router uses Serial 0/1 more because there is primarily only traffic from one source-destination pair (flow) passing. This traffic flow will use the same path through Serial 0/1.

Example 6-17. Uneven Bucket Distribution with One Primary User

```
Code View: Scroll / Show All  
R1#show ip cef 10.1.5.1 internal  
10.1.5.1/32, version 30, epoch 0, per-destination sharing  
0 packets, 0 bytes  
via 10.1.2.2, Serial0/0, 0 dependencies  
  traffic share 1  
  next hop 10.1.2.2, Serial0/0
```

valid adjacency

via 10.1.2.10, Serial0/2, 0 dependencies

traffic share 1

next hop 10.1.2.10, Serial0/2

valid adjacency

via 10.1.2.6, Serial0/1, 0 dependencies

traffic share 1

next hop 10.1.2.6, Serial0/1

valid adjacency

via 10.1.2.14, Serial0/3, 0 dependencies

traffic share 1

next hop 10.1.2.14, Serial0/3

valid adjacency

0 packets, 0 bytes switched through the prefix

tmstats: external 0 packets, 0 bytes

internal 0 packets, 0 bytes

Load distribution: 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 (refcount 1)

Hash	OK	Interface	Address	Packets
1	Y	Serial0/0	point2point	0
2	Y	Serial0/2	point2point	0
3	Y	Serial0/1	point2point	62526
4	Y	Serial0/3	point2point	0
5	Y	Serial0/0	point2point	0
6	Y	Serial0/2	point2point	0
7	Y	Serial0/1	point2point	0
8	Y	Serial0/3	point2point	1887

9	Y	Serial0/0	point2point	0
10	Y	Serial0/2	point2point	0
11	Y	Serial0/1	point2point	0
12	Y	Serial0/3	point2point	0
13	Y	Serial0/0	point2point	0
14	Y	Serial0/2	point2point	0
15	Y	Serial0/1	point2point	0
16	Y	Serial0/3	point2point	0

Here is another example where all traffic is primarily taking one interface outbound. Refer to Figure 6-5 again. In this case, there are numerous different sources all going to destination 10.1.5.1. Why is traffic distribution not equal with CEF per-destination as seen before in Example 6-12? From the output in Example 6-18, it appears that traffic is primarily taking interface Serial 0/3 outbound.

Example 6-18. Verifying Traffic Distribution with Numerous Flows and CEF Per-Destination

Code View: Scroll / Show All

R1#show interfaces

!Output omitted for brevity

Serial0/0 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.1/30

!Output omitted for brevity

30 second output rate 0 bits/sec, 0 packets/sec

114 packets input, 7024 bytes, 0 no buffer

Received 14 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

38 packets output, 2468 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/1 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.5/30

!Output omitted for brevity

30 second output rate 0 bits/sec, 0 packets/sec

84 packets input, 5228 bytes, 0 no buffer

Received 14 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

39 packets output, 2532 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/2 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.9/30

!Output omitted for brevity

30 second output rate 0 bits/sec, 0 packets/sec

91 packets input, 5652 bytes, 0 no buffer

Received 14 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

41 packets output, 2620 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Serial0/3 is up, line protocol is up

Hardware is PowerQUICC Serial

Internet address is 10.1.2.13/30

!Output omitted for brevity

30 second output rate 26000 bits/sec, 58 packets/sec

103 packets input, 6376 bytes, 0 no buffer

Received 14 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

6883 packets output, 391984 bytes, 0 underruns

0 output errors, 0 collisions, 0 interface resets

0 output buffer failures, 0 output buffers swapped out

0 carrier transitions

DCD=up DSR=up DTR=up RTS=up CTS=up

Checking the routing table, as seen in Example 6-19, there are four equal-cost paths to 10.1.5.1.

Example 6-19. Verifying Equal-Cost Paths with CEF Per-Destination

Code View: Scroll / Show All

R1#show ip route 10.1.5.1

Routing entry for 10.1.5.1/32

Known via "eigrp 100", distance 90, metric 2300416, type internal

Redistributing via eigrp 100

Last update from 10.1.2.2 on Serial0/0, 2w0d ago

Routing Descriptor Blocks:

10.1.2.14, from 10.1.2.14, 2w0d ago, via Serial0/3

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 2/255, Hops 2

10.1.2.6, from 10.1.2.6, 2w0d ago, via Serial0/1

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 3/255, Hops 2

* 10.1.2.10, from 10.1.2.10, 2w0d ago, via Serial0/2

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 3/255, Hops 2

10.1.2.2, from 10.1.2.2, 2w0d ago, via Serial0/0

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 3/255, Hops 2

The CEF FIB entry also looks correct, as seen in Example 6-20. Per-destination load sharing is in operation, and four paths match the routing table.

Example 6-20. Verifying CEF Entry for 10.1.5.1

Code View: Scroll / Show All

```
R1#show ip cef 10.1.5.1
10.1.5.1/32, version 32, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 1
    next hop 10.1.2.6, Serial0/1
    valid adjacency
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 1
    next hop 10.1.2.2, Serial0/0
    valid adjacency
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes
```

The adjacency is also correct for a point-to-point interface. As shown in Example 6-21, the adjacency is point2point for the serial interfaces. For a broadcast interface, you could also check the source and destination MAC address with the show adjacency <interface> detail command.

Example 6-21. Verifying the CEF Adjacency

Code View: Scroll / Show All

R1#show adjacency serial0/3 detail

Protocol	Interface	Address
IP	Serial0/3	point2point(35)
		0 packets, 0 bytes
		0F000800
		CEF expires: 00:02:42
		refresh: 00:00:42
		Epoch: 0

R1#show adjacency serial0/2 detail

Protocol	Interface	Address
IP	Serial0/2	point2point(35)
		0 packets, 0 bytes
		0F000800
		CEF expires: 00:02:36
		refresh: 00:00:36
		Epoch: 0

R1#show adjacency serial0/1 detail

Protocol	Interface	Address
IP	Serial0/1	point2point(35)
		0 packets, 0 bytes
		0F000800
		CEF expires: 00:02:32
		refresh: 00:00:32
		Epoch: 0

R1#show adjacency serial0/0 detail

Protocol	Interface	Address
IP	Serial0/0	point2point(35)


```
0 packets, 0 bytes
0F000800
CEF expires: 00:02:24
refresh: 00:00:24
Epoch: 0
```

CEF outputs are as expected so far, although traffic is primarily taking one link. It is now important to check whether CEF is passing traffic to fast-switching or processing-switching levels because it cannot handle the traffic at the CEF switching level. The `show cef not` command is unsupported and receive counters are incrementing, as shown in the output in Example 6-22. In later code, the `show ip cef switching statistics` command gives detailed information about why a pass occurs and replaces the `show cef not` command.

Example 6-22. Verifying CEF Passing Packets to Other Switching Path

Code View: Scroll / Show All

```
R1#show cef not
```

```
CEF Packets passed on to next switching layer
```

Slot	No_adj	No_encap	Unsup'ted	Redirect	Receive	Options	Access	Frag
RP	0	0	743	0	334	0	0	0

```
R1#show cef not
```

```
CEF Packets passed on to next switching layer
```

Slot	No_adj	No_encap	Unsup'ted	Redirect	Receive	Options	Access	Frag
RP	0	0	852	0	382	0	0	0

The important piece is that the unsupported counters are incrementing. Therefore, CEF is passing traffic to the fast-switching and possibly the process-switching level. When checking the fast cache in Example 6-23, there is a fast cache entry for 10.1.5.1.

Example 6-23. Verifying an Empty Fast Cache with CEF Enabled

```

R1#show ip cache

IP routing cache 1 entry, 180 bytes

 8 adds, 7 invalidates, 0 refcounts

Minimum invalidation interval 2 seconds, maximum interval 5 seconds,
  quiet interval 3 seconds, threshold 0 requests

Invalidation rate 0 in last second, 0 in last 3 seconds

Last full cache invalidation occurred 00:19:11 ago

Prefix/Length    Age    Interface    Next Hop
10.1.5.1/32     00:01:56 Serial0/3    10.1.2.14

R1#

```

A fast-cache entry should not appear in the output if CEF is switching the packet. The traffic flow enters Router R1 through interface Fast Ethernet 0/0. The inbound interface Fast Ethernet 0/0 on Router R1 in Figure 6-5 has CEF disabled, as seen in Example 6-24.

Example 6-24. Verifying CEF Operation on the Inbound Interface

```

Code View: Scroll / Show All

R1#show cef interface fast0/0

FastEthernet0/0 is up (if_number 4)

  Corresponding hwidb fast_if_number 4

  Corresponding hwidb firstsw->if_number 4

Internet address is 10.1.1.2/24

ICMP redirects are always sent

Per packet load-sharing is disabled

IP unicast RPF check is disabled

Inbound access list is not set

Outbound access list is not set

IP policy routing is disabled

```

```
BGP based policy accounting is disabled
Hardware idb is FastEthernet0/0
Fast switching type 1, interface type 18
IP CEF switching disabled
IP Fast switching turbo vector
Input fast flags 0x0, Output fast flags 0x0
ifindex 2(2)
Slot 0 Slot unit 0 Unit 0 VC -1
Transmit limit accumulator 0x0 (0x0)
IP MTU 1500
R1#
```

The issue is clear after verifying the configuration on Fast Ethernet 0/0, as shown in Example 6-25. The router has CEF enabled globally but not under the interface.

Example 6-25. Verifying CEF Configuration on the Inbound Interface

```
R1# show run interface fast0/0
Building configuration...

Current configuration : 154 bytes
!
interface FastEthernet0/0
 ip address 10.1.1.2 255.255.255.0
 no ip route-cache cef
 load-interval 30
 speed 100
 full-duplex
```

```
no cns route-cache
end
```

Even after enabling CEF globally, it is important to check that CEF is operational on the interface level and especially on the inbound interface. Check this using the `show cef interface` command. If CEF is not operational on the inbound interface, the packet will not be CEF switched. In this case, to reenable CEF on Fast Ethernet 0/0, just enter the `ip route-cache cef` command under the interface. In Example 6-26, each serial interface is now passing traffic outbound.

Example 6-26. Verifying Load Distribution After Enabling CEF on the Inbound Interface

Code View: Scroll / Show All

```
R1#show interfaces | include Serial | output drops | Output queue | output rate
```

```
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

```
Output queue: 0/40 (size/max)
```

```
30 second output rate 1000 bits/sec, 2 packets/sec
```

```
Serial0/0 is up, line protocol is up
```

```
Hardware is PowerQUICC Serial
```

```
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

```
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
```

```
30 second output rate 8000 bits/sec, 15 packets/sec
```

```
Serial0/1 is up, line protocol is up
```

```
Hardware is PowerQUICC Serial
```

```
Input queue: 1/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

```
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
```

```
30 second output rate 8000 bits/sec, 20 packets/sec
```

```
Serial0/2 is up, line protocol is up
```

```
Hardware is PowerQUICC Serial
```

```
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
```

```
Output queue: 0/1000/64/0 (size/max total/threshold/drops)
```

```
30 second output rate 5000 bits/sec, 11 packets/sec
```

Serial0/3 is up, line protocol is up

Hardware is PowerQUICC Serial

Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0

Output queue: 0/1000/64/0 (size/max total/threshold/drops)

30 second output rate 7000 bits/sec, 14 packets/sec

Originally, CEF did not support many features in its switching path. In addition, many Cisco platforms did not run CEF. Now, CEF is the default switching path in most Cisco IOS versions and platforms. If a Cisco router does not support a feature in the CEF path, it must punt the packet to the next switching path. Usually, the router passes to fast switching and then to process switching.

Follow the same troubleshooting steps used when the inbound interface Fast Ethernet 0/0 did not have CEF enabled in Examples 6-18 through Example 6-26. Therefore, if packets are primarily taking one path, CEF might not support a feature on the inbound interface. For example, enabling software compression on Serial 0/0, Serial 0/1, Serial 0/2, and Serial 0/3 disables CEF on those interfaces in Router R1. Router R1 is running Cisco IOS Release 12.3(10a). Example 6-27 that shows CEF is not operational on these interfaces. Therefore, traffic entering these interfaces will not be CEF switched to the outbound interface Fast Ethernet 0/0.

Example 6-27. Verifying CEF on an Interface with Software Compression

Code View: Scroll / Show All

```
R1#show cef interface
```

!Output omitted for brevity

Serial0/0 is up (if_number 5)

Corresponding hwidb fast_if_number 5

Corresponding hwidb firstsw->if_number 5

Internet address is 10.1.2.1/30

!Output omitted for brevity

Packets switched to this interface are dropped to the next slow path: Compression

Hardware idb is Serial0/0

Fast switching type 4, interface type 60

IP CEF switching disabled

IP Null turbo vector

Input fast flags 0x0, Output fast flags 0x0

ifindex 3(3)

Slot 0 Slot unit 0 Unit 0 VC -1

Transmit limit accumulator 0x0 (0x0)

IP MTU 1500

Serial0/1 is up (if_number 6)

Corresponding hwidb fast_if_number 6

Corresponding hwidb firstsw->if_number 6

Internet address is 10.1.2.5/30

!Output omitted for brevity

Packets switched to this interface are dropped to the next slow path: Compression

Hardware idb is Serial0/1

Fast switching type 4, interface type 60

IP CEF switching disabled

IP Null turbo vector

Input fast flags 0x0, Output fast flags 0x0

ifindex 4(4)

Slot 0 Slot unit 1 Unit 1 VC -1

Transmit limit accumulator 0x0 (0x0)

IP MTU 1500

Serial0/2 is up (if_number 7)

Corresponding hwidb fast_if_number 7

Corresponding hwidb firstsw->if_number 7

Internet address is 10.1.2.9/30

!Output omitted for brevity

Packets switched to this interface are dropped to the next slow path: Compression

Hardware idb is Serial0/2

```
Fast switching type 4, interface type 60
IP CEF switching disabled
IP Null turbo vector
Input fast flags 0x0, Output fast flags 0x0
ifindex 5(5)
Slot 0 Slot unit 2 Unit 2 VC -1
Transmit limit accumulator 0x0 (0x0)
IP MTU 1500
Serial0/3 is up (if_number 8)
Corresponding hwidb fast_if_number 8
Corresponding hwidb firstsw->if_number 8
Internet address is 10.1.2.13/30
!Output omitted for brevity
Packets switched to this interface are dropped to the next slow path: Compression
Hardware idb is Serial0/3
Fast switching type 4, interface type 60
IP CEF switching disabled
IP Null turbo vector
Input fast flags 0x0, Output fast flags 0x0
ifindex 6(6)
Slot 0 Slot unit 3 Unit 3 VC -1
Transmit limit accumulator 0x0 (0x0)
IP MTU 1500
```

Sometimes a network administrator can notice that not all the paths are in use. Perhaps only three out of four equal-cost links have a load. Before blaming the issue on CEF, always check the routing table. In this case, when checking the routing table, only three paths are known to 10.1.5.1 instead of four, as shown in Example 6-28.

Example 6-28. Verifying the Routing Table for All Paths

```
R1#show ip route 10.1.5.1

Routing entry for 10.1.5.1/32

  Known via "eigrp 100", distance 90, metric 2300416, type internal

  Redistributing via eigrp 100

  Last update from 10.1.2.10 on Serial0/2, 00:03:04 ago

  Routing Descriptor Blocks:

    * 10.1.2.14, from 10.1.2.14, 00:03:04 ago, via Serial0/3

      Route metric is 2300416, traffic share count is 1

      Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

      Reliability 255/255, minimum MTU 1500 bytes

      Loading 1/255, Hops 2

    10.1.2.6, from 10.1.2.6, 00:03:04 ago, via Serial0/1

      Route metric is 2300416, traffic share count is 1

      Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

      Reliability 255/255, minimum MTU 1500 bytes

      Loading 1/255, Hops 2

    10.1.2.10, from 10.1.2.10, 00:03:04 ago, via Serial0/2

      Route metric is 2300416, traffic share count is 1

      Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

      Reliability 255/255, minimum MTU 1500 bytes

      Loading 1/255, Hops 2
```

Interface Serial 0/0 is not in the listed possible paths to reach 10.1.5.1. Looking at the configuration for interface Serial 0/0 in Example 6-29, you can see that the bandwidth is set to 1543 kbps while the other interfaces have 1544 kbps configured.

Example 6-29. Verifying Serial 0/0 Configuration

```
R1#show run interface Serial0/0

Building configuration...
```



```
Current configuration : 131 bytes
```

```
!
```

```
interface Serial0/0
```

```
bandwidth 1543
```

```
ip address 10.1.2.1 255.255.255.252
```

```
load-interval 30
```

```
fair-queue
```

```
no clns route-cache
```

```
end
```

EIGRP uses bandwidth and delay by default to define its metric. Therefore, the lower metric existed for interfaces Serial 0/3, Serial 0/1, and Serial 0/2 than for interface Serial 0/0. Example 6-30 shows the composite metric is 2301440 through Serial 0/0 versus 2300416 through the other interfaces. Changing the bandwidth to the desired value of 1544 kbps on interface Serial 0/0 resolves this issue. Therefore, load sharing depends on the correct routes being in the routing table.

Example 6-30. Verifying EIGRP Metric for 10.1.5.1 Destination

```
Code View: Scroll / Show All
```

```
R1#show ip eigrp top 10.1.5.1 255.255.255.255
```

```
IP-EIGRP (AS 100): Topology entry for 10.1.5.1/32
```

```
State is Passive, Query origin flag is 1, 3 Successor(s), FD is 2300416
```

```
Routing Descriptor Blocks:
```

```
10.1.2.10 (Serial0/2), from 10.1.2.10, Send flag is 0x0
```

```
Composite metric is (2300416/156160), Route is Internal
```

```
Vector metric:
```

```
Minimum bandwidth is 1544 Kbit
```

```
Total delay is 25100 microseconds
```

```
Reliability is 255/255
```

```
Load is 1/255
```

```
Minimum MTU is 1500
```

Hop count is 2

10.1.2.6 (Serial0/1), from 10.1.2.6, Send flag is 0x0

Composite metric is (2300416/156160), Route is Internal

Vector metric:

Minimum bandwidth is 1544 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

10.1.2.14 (Serial0/3), from 10.1.2.14, Send flag is 0x0

Composite metric is (2300416/156160), Route is Internal

Vector metric:

Minimum bandwidth is 1544 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

10.1.2.2 (Serial0/0), from 10.1.2.2, Send flag is 0x0

Composite metric is (2301440/156160), Route is Internal

Vector metric:

Minimum bandwidth is 1543 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

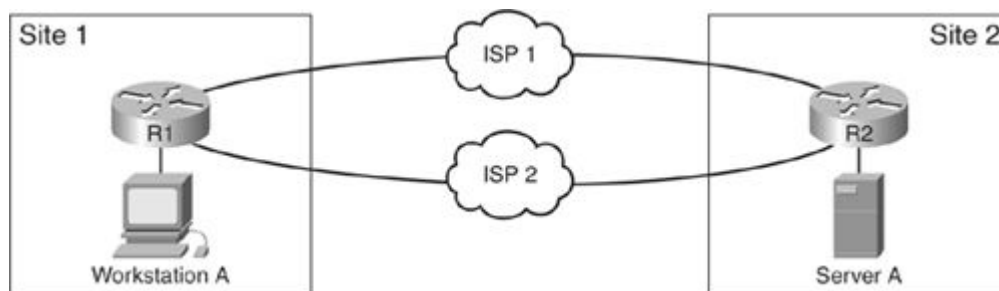


CEF Per-Packet Load Sharing Causing Performance Issues

Sometimes a particular design is not suitable for CEF per-packet load sharing. For example, in Figure 6-6, a TCP-based FTP session going across two WAN circuits by two different service providers can have issues with per-packet load sharing.

Figure 6-6. Per-Packet Load Sharing Across Two Different Service Providers

[View full size image]



When a sender generates a TCP stream, it sends packets in a certain numbered order. When the destination receives the packets in a different order, this is out of order. With per-packet load sharing as seen in Figure 6-6, packets can be received out of order at the destination. The destination application has to reorder the packets, and this can cause TCP to request "missing" packets (retransmissions), to send duplicate acknowledgments, and to reduce the TCP window size. Hence, the application throughput and performance will suffer.

With per-packet load sharing and heavy load, TCP throttling of data can cause utilization of only a fraction of the available bandwidth. TCP uses a sliding window concept to handle flow control and to try to achieve efficient data transmission. The receiver sends the window size to the sender in a TCP session. The window size tells the sender how much data to send before expecting an acknowledgment. In theory, an optimally sized window allows a session to send as fast as the network can accept. However, retransmissions, duplicate acknowledgments, and latency will affect the size of the TCP window and cause it to decrease. TCP tries to detect congestion by monitoring these characteristics and reduces the window size based on behavior. Utilizing policy-based routing and other load-sharing techniques can improve application throughput in this layout.

Also, in a CEF per-packet scenario, if a Cisco router does not support a feature in the CEF path, it must pass the packet to the next switching path. Therefore, fast switching might be handling the packets with CEF per-packet enabled. Detection of this scenario is easy because all packets will appear to take one path only for a specific destination. A fast-cache entry will also exist.

In another case, CEF might pass to fast switching and then fast switching might pass to process switching. In this case, CEF per-packet appears to be working properly by looking at the utilization. However, the `show cef not-cef-switched` command output will show that the router is passing the packets onto the slower switching path. The output from the `show interfaces stats` command will show that packets are taking the process path instead of the faster option. For example, in certain Cisco IOS versions, if options are set in the IP header of a

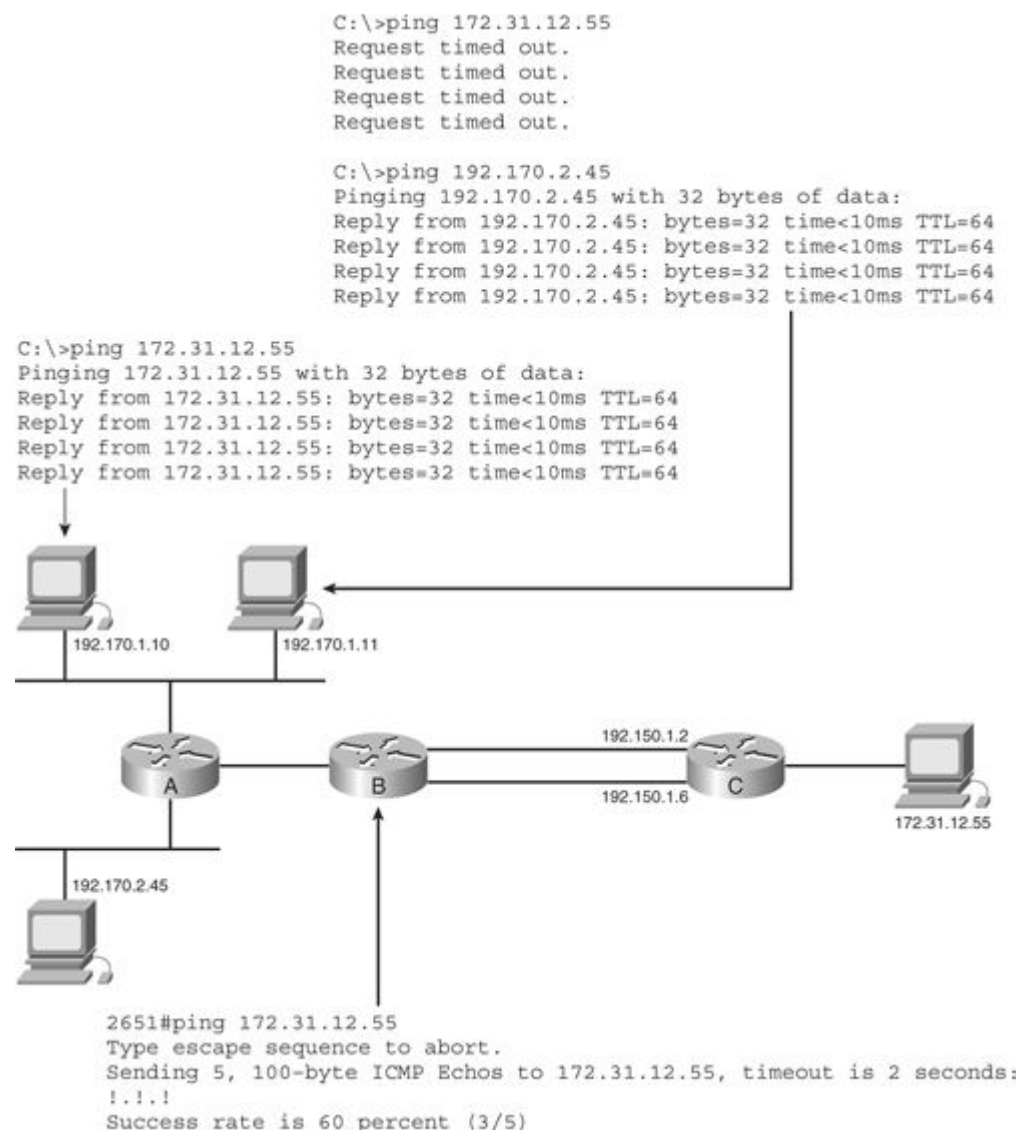
packet, the router process switches each packet. Therefore, understanding the type of traffic passing through the router, the router's configuration, and feature support is important in troubleshooting this behavior.

Troubleshooting a Single-Path Failure with CEF Load Sharing

Single-path failures in a load-sharing environment, while not necessarily caused by a problem in CEF itself, are often frustrating to troubleshoot when you are running CEF. Figure 6-7 illustrates a small network in which a single-path failure has occurred.

Figure 6-7. Single-Path Failure with CEF Load Sharing

[View full size image]



From host 192.170.1.10, it appears that connectivity to the server, 172.31.12.55, is just fine. However, from host 192.170.1.11, the server is not reachable. At first, this would appear to be a problem with the host itself, but 192.170.1.11 can reach other servers that are not on the 192.170.1.0 segment. For example, 192.170.1.11 can reach 192.170.2.45 but it cannot reach server 172.31.12.55. The next step beyond ruling out a host problem is to then step through the path, attempting to discover where the problem occurs.

In this case, pinging from Router A shows that 172.31.12.55 is reachable, but pinging from Router B shows that it is only reachable 50 percent of the time. This information does not seem to provide any clues about where the problem lies, but if you consider the way CEF load-shares, and combine that with the way routers switch packets that are locally generated, it all points to one problem.

First, recall that CEF load-shares based on a hash of the source and destination address. All the traffic traveling to 172.31.12.55 and originating from 192.170.1.10 will pass along one of the two parallel paths between Routers B and C. All the traffic traveling to 172.31.12.55, and originating from 192.170.1.11, will pass along a single path between Routers B and C as well. Assume that these two hosts' traffic will pass along different paths between Routers B and C. This then explains how one host could reach 172.31.12.55 and the other cannot. If only one path out of a set of parallel paths has failed, some hosts will be able to traverse one of the other (working) links, while other hosts' traffic will fail on the problematic link.

However, why does Router B reach 172.31.12.55 with 50 percent of its packets? When a router generates packets locally, such as routing protocol updates or pings, it will process-switch those packets. Process-switched packets are load shared on a per-packet basis. In this case, only two paths are in the routing table, so every other packet will take the alternate link. When pinging from a router, if one link among several parallel links has failed, the percentage of pings that fail will be proportional to the number of links available.

There are two ways to verify which link is problematic. The first way is to ping between the two routers that connect through parallel links using the extended options in the ping command to force the ping to take a particular link, as shown in Example 6-31.

Example 6-31. Verifying a Problematic Link Through Pings

```
router#ping
Protocol [ip]:
Target IP address: 192.160.11.11
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: serial0/2
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 192.160.11.11, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms

You can thus force the source interface to be one of the two parallel paths and test each one separately. Make certain that you are pinging just end to end and not to some remote destination. In this case, you want to set the destination address to the other side of the serial link.

Another option is to use the show ip cef exact-route command, as follows:

```
router#show ip cef exact-route 192.168.1.11 172.31.12.55
```

```
192.168.1.11 -> 172.31.12.55 : Serial0/3
```

Using the show ip cef exact-route command on the router with the parallel links attached, you can determine which path the packets sourced from 192.168.1.11 and destined to 172.31.12.55 will take and focus on troubleshooting just that link.

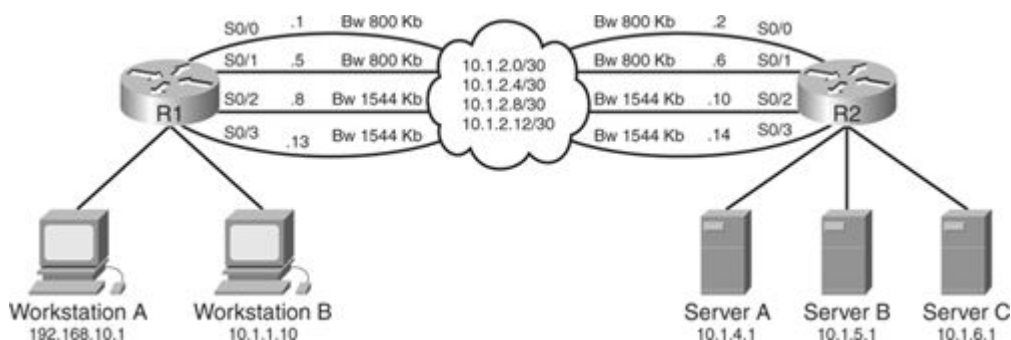
CEF Traffic-Share Allocation

EIGRP can install unequal-cost paths in the routing table. When the routing table installs unequal-cost routes, the traffic-share value is also set to weight some paths more than other paths. This traffic-share value will be carried to CEF also and used to create the load distribution table. According to the load distribution table, CEF assigns more sessions to the heavier-weighted paths to use them more than other paths in the instance of unequal-cost route installation. This works with CEF per-destination and CEF per-packet scenarios. In the unequal-cost per-packet scenario, CEF will still send traffic in a round-robin fashion, but the router will send packets on a weighted path multiple times before moving to the next path.

In Figure 6-8, there are four paths between 192.168.10.1 and 10.1.5.1. EIGRP unequal load sharing can utilize all four paths between the two destinations. Without EIGRP, other routing protocols would only choose the bottom two paths through Serial 0/2 and Serial 0/3 to reach 10.1.5.1.

Figure 6-8. Network Requiring EIGRP Unequal Load Sharing

[View full size image]



In Example 6-32, notice that without an unequal load-sharing configuration, Router R1 only installs two paths to reach 10.1.5.1 with regular EIGRP configuration.

Example 6-32. Routing Table of Router R1 for Destination 10.1.5.1 with Normal EIGRP Configuration

```
R1#show ip route 10.1.5.1
Routing entry for 10.1.5.1/32
  Known via "eigrp 100", distance 90, metric 2300416, type internal
  Redistributing via eigrp 100
  Last update from 10.1.2.10 on Serial0/2, 00:28:02 ago
  Routing Descriptor Blocks:
  * 10.1.2.14, from 10.1.2.14, 00:28:02 ago, via Serial0/3
    Route metric is 2300416, traffic share count is 1
    Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 2
  10.1.2.10, from 10.1.2.10, 00:28:02 ago, via Serial0/2
    Route metric is 2300416, traffic share count is 1
    Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 2
```

In EIGRP, a path must be a feasible successor for it to be available for load sharing. A path is a feasible successor when the reported distance is less than the feasible distance. Looking at the EIGRP topology in Example 6-33, unequal load sharing is possible because the feasibility condition is true. For paths through Serial 0/0 and Serial 0/1, the reported distance of 156160 is less than the feasible distance of 2300416.

Example 6-33. Checking Feasibility Condition for Other Paths to 10.1.5.1 on Router R1

```
Code View: Scroll / Show All
R1#show ip eigrp topology 10.1.5.1 255.255.255.255
IP-EIGRP (AS 100): Topology entry for 10.1.5.1/32
```

State is Passive, Query origin flag is 1, 2 Successor(s), FD is 2300416

Routing Descriptor Blocks:

10.1.2.10 (Serial0/2), from 10.1.2.10, Send flag is 0x0

Composite metric is (2300416/156160), Route is Internal

Vector metric:

Minimum bandwidth is 1544 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

10.1.2.14 (Serial0/3), from 10.1.2.14, Send flag is 0x0

Composite metric is (2300416/156160), Route is Internal

Vector metric:

Minimum bandwidth is 1544 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

10.1.2.6 (Serial0/1), from 10.1.2.6, Send flag is 0x0

Composite metric is (3842560/156160), Route is Internal

Vector metric:

Minimum bandwidth is 800 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 2/255

Minimum MTU is 1500

Hop count is 2

10.1.2.2 (Serial0/0), from 10.1.2.2, Send flag is 0x0

Composite metric is (3842560/156160), Route is Internal

Vector metric:

Minimum bandwidth is 800 Kbit

Total delay is 25100 microseconds

Reliability is 255/255

Load is 1/255

Minimum MTU is 1500

Hop count is 2

R1#

The EIGRP variance <multiplier> command instructs the router to include routes with a metric less than the multiplier times the minimum metric route for that destination. The metric of 3842560 for the Serial 0/0 and Serial 0/1 path is less than two times the minimum metric of 2300416 for 10.5.1.1. So, setting the variance multiplier to 2 under EIGRP on Router R1 will allow all four paths in the routing table:

```
R1(config)#router eigrp 100
```

```
R1(config-router)#variance 2
```

After configuration, there are now four paths to reach 10.1.5.1 in the routing table of Router R1, as shown in Example 6-34.

Example 6-34. Verifying Router R1's Routing Table After Applying Variance

Code View: Scroll / Show All

```
R1#show ip route 10.1.5.1
```

```
Routing entry for 10.1.5.1/32
```

```
Known via "eigrp 100", distance 90, metric 2300416, type internal
```

```
Redistributing via eigrp 100
```

Last update from 10.1.2.2 on Serial0/0, 00:00:52 ago

Routing Descriptor Blocks:

* 10.1.2.14, from 10.1.2.14, 00:00:52 ago, via Serial0/3

Route metric is 2300416, traffic share count is 5

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 1/255, Hops 2

10.1.2.10, from 10.1.2.10, 00:00:52 ago, via Serial0/2

Route metric is 2300416, traffic share count is 5

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 1/255, Hops 2

10.1.2.6, from 10.1.2.6, 00:00:52 ago, via Serial0/1

Route metric is 3842560, traffic share count is 3

Total delay is 25100 microseconds, minimum bandwidth is 800 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 2/255, Hops 2

10.1.2.2, from 10.1.2.2, 00:00:53 ago, via Serial0/0

Route metric is 3842560, traffic share count is 3

Total delay is 25100 microseconds, minimum bandwidth is 800 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 1/255, Hops 2

Note that the traffic-share value in Example 6-34 also takes into account the difference in the metrics across the multiple paths. The traffic-share count for the 1544-kbps paths is five while the other paths have three. The CEF load distribution table also shows the change in Example 6-35. Notice that buckets 1–12 equally utilize each path three times. Buckets 13–16 utilize paths Serial 0/2 and Serial 0/3 two more times each. Therefore, the router uses paths Serial 0/2 and Serial 0/3 five times and Serial 0/0 and Serial 0/1 three times.

Example 6-35. Verifying the Load Distribution Table

Code View: Scroll / Show All

```
R1#show ip cef 10.1.5.1 internal
```

```
10.1.5.1/32, version 71, epoch 0, per-destination sharing
```

```
0 packets, 0 bytes
```

```
via 10.1.2.14, Serial0/3, 0 dependencies
```

```
traffic share 5
```

```
next hop 10.1.2.14, Serial0/3
```

```
valid adjacency
```

```
via 10.1.2.10, Serial0/2, 0 dependencies
```

```
traffic share 5
```

```
next hop 10.1.2.10, Serial0/2
```

```
valid adjacency
```

```
via 10.1.2.6, Serial0/1, 0 dependencies
```

```
traffic share 3
```

```
next hop 10.1.2.6, Serial0/1
```

```
valid adjacency
```

```
via 10.1.2.2, Serial0/0, 0 dependencies
```

```
traffic share 3
```

```
next hop 10.1.2.2, Serial0/0
```

```
valid adjacency
```

```
0 packets, 0 bytes switched through the prefix
```

```
tmstats: external 0 packets, 0 bytes
```

```
internal 0 packets, 0 bytes
```

```
Load distribution: 0 1 2 3 0 1 2 3 0 1 2 3 0 1 0 1 (refcount 1)
```

Hash	OK	Interface	Address	Packets
------	----	-----------	---------	---------

```
1 Y Serial0/3      point2point  1200
2 Y Serial0/2      point2point   774
3 Y Serial0/1      point2point  1847
4 Y Serial0/0      point2point   384
5 Y Serial0/3      point2point  1153
6 Y Serial0/2      point2point  2387
7 Y Serial0/1      point2point  1173
8 Y Serial0/0      point2point  1962
9 Y Serial0/3      point2point  2913
10 Y Serial0/2     point2point  2333
11 Y Serial0/1     point2point   781
12 Y Serial0/0     point2point  2814
13 Y Serial0/3     point2point  1182
14 Y Serial0/2     point2point   385
15 Y Serial0/3     point2point  2026
16 Y Serial0/2     point2point  1600

R1#
```

With EIGRP, you can also change how EIGRP sets the traffic-share value with the traffic-share command. By default, the command uses the balanced option. You can also set it to use only the minimum metric paths. This is identical to the forwarding behavior without the variance command. However, with the traffic-share min across-interfaces command and the variance command, the routing table will install all feasible paths, which decreases convergence time.

In Example 6-36, notice that all the paths are in the routing table with the traffic-share min across-interfaces configuration. However, only the paths through Serial 0/3 and Serial 0/2 have a nonzero traffic-share count. If the traffic-share count is 0, the path is not in use. However, if the primary paths through Serial 0/3 and Serial 0/2 go away, the other two paths will be in use and have a nonzero traffic-share count.

Example 6-36. show ip route Output with traffic-share min across-interfaces Configuration

```
Code View: Scroll / Show All
```

```
R1#show ip route 10.1.5.1
```

```
Routing entry for 10.1.5.1/32
```

```
Known via "eigrp 100", distance 90, metric 2300416, type internal
```

```
Redistributing via eigrp 100
```

```
Last update from 10.1.2.2 on Serial0/0, 00:03:30 ago
```

```
Routing Descriptor Blocks:
```

```
10.1.2.14, from 10.1.2.14, 00:03:30 ago, via Serial0/3
```

```
Route metric is 2300416, traffic share count is 1
```

```
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 1/255, Hops 2
```

```
* 10.1.2.10, from 10.1.2.10, 00:03:30 ago, via Serial0/2
```

```
Route metric is 2300416, traffic share count is 1
```

```
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 1/255, Hops 2
```

```
10.1.2.6, from 10.1.2.6, 00:03:30 ago, via Serial0/1
```

```
Route metric is 3842560, traffic share count is 0
```

```
Total delay is 25100 microseconds, minimum bandwidth is 800 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 2/255, Hops 2
```

```
10.1.2.2, from 10.1.2.2, 00:03:32 ago, via Serial0/0
```

```
Route metric is 3842560, traffic share count is 0
```

```
Total delay is 25100 microseconds, minimum bandwidth is 800 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 1/255, Hops 2
```

```
R1#
```

In Example 6-37, the load distribution table includes only path 2 (Serial 0/2) and path 3 (Serial 0/3).

Example 6-37. CEF Load Distribution with traffic-share min across-interfaces Configuration

Code View: Scroll / Show All

```
R1#show ip cef 10.1.5.1 internal
10.1.5.1/32, version 126, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.1.2.6, Serial0/1, 0 dependencies
    traffic share 0
    next hop 10.1.2.6, Serial0/1
    valid adjacency
  via 10.1.2.2, Serial0/0, 0 dependencies
    traffic share 0
    next hop 10.1.2.2, Serial0/0
    valid adjacency
  via 10.1.2.10, Serial0/2, 0 dependencies
    traffic share 1
    next hop 10.1.2.10, Serial0/2
    valid adjacency
  via 10.1.2.14, Serial0/3, 0 dependencies
    traffic share 1
    next hop 10.1.2.14, Serial0/3
    valid adjacency

0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
```

internal 0 packets, 0 bytes

Load distribution: 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 (refcount 1)

Hash	OK	Interface	Address	Packets
1	Y	Serial0/2	point2point	638
2	Y	Serial0/3	point2point	393
3	Y	Serial0/2	point2point	1024
4	Y	Serial0/3	point2point	235
5	Y	Serial0/2	point2point	602
6	Y	Serial0/3	point2point	1349
7	Y	Serial0/2	point2point	617
8	Y	Serial0/3	point2point	1121
9	Y	Serial0/2	point2point	1532
10	Y	Serial0/3	point2point	1300
11	Y	Serial0/2	point2point	409
12	Y	Serial0/3	point2point	1580
13	Y	Serial0/2	point2point	648
14	Y	Serial0/3	point2point	224
15	Y	Serial0/2	point2point	1093
16	Y	Serial0/3	point2point	801

R1#

In the case of equal-cost paths, CEF does a check to verify whether the number of buckets is equally divisible by the number of paths. If not, CEF disables the last remaining buckets. For example, if there are six equal-cost paths, the first 12 buckets utilize each path two times. Then, CEF disables the remaining four buckets.

Table 6-6 shows the load distribution table for various numbers of equal-cost paths. CEF per-packet and CEF per-destination utilize the traffic-share allocation to control packet distribution.

Table 6-6. CEF Load-Share Distribution Bucket Usage with Multiple Paths

Number of Equal-Cost Paths	Load-Share Distribution Table by Bucket															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	—
4	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
5	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	—
6	0	1	2	3	4	5	0	1	2	3	4	5	—	—	—	—
7	0	1	2	3	4	5	6	0	1	2	3	4	5	6	—	—
8	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

In Example 6-38, notice that for three equal-cost paths to reach destination 10.1.5.1, there are only 15 hash buckets used instead of 16, as seen with the four equal-cost paths in Example 6-13.

Example 6-38. Only 15 Hash Buckets Are Used with Three Equal-Cost Paths

Code View: Scroll / Show All

```
R1#show ip route 10.1.5.1
```

```
Routing entry for 10.1.5.1/32
```

```
Known via "eigrp 100", distance 90, metric 2300416, type internal
```

```
Redistributing via eigrp 100
```

```
Last update from 10.1.2.6 on Serial0/1, 00:05:10 ago
```

```
Routing Descriptor Blocks:
```

```
* 10.1.2.2, from 10.1.2.2, 00:05:10 ago, via Serial0/0
```

```
Route metric is 2300416, traffic share count is 1
```

```
Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```


Loading 1/255, Hops 2

10.1.2.10, from 10.1.2.10, 00:05:10 ago, via Serial0/2

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 1/255, Hops 2

10.1.2.6, from 10.1.2.6, 00:05:10 ago, via Serial0/1

Route metric is 2300416, traffic share count is 1

Total delay is 25100 microseconds, minimum bandwidth is 1544 Kbit

Reliability 255/255, minimum MTU 1500 bytes

Loading 1/255, Hops 2

R1#show ip cef 10.1.5.1 internal

10.1.5.1/32, version 26, epoch 0, per-destination sharing

0 packets, 0 bytes

via 10.1.2.2, Serial0/0, 0 dependencies

traffic share 1

next hop 10.1.2.2, Serial0/0

valid adjacency

via 10.1.2.10, Serial0/2, 0 dependencies

traffic share 1

next hop 10.1.2.10, Serial0/2

valid adjacency

via 10.1.2.6, Serial0/1, 0 dependencies

traffic share 1

next hop 10.1.2.6, Serial0/1

valid adjacency

0 packets, 0 bytes switched through the prefix

tmstats: external 0 packets, 0 bytes

internal 0 packets, 0 bytes

Load distribution: 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 (refcount 1)

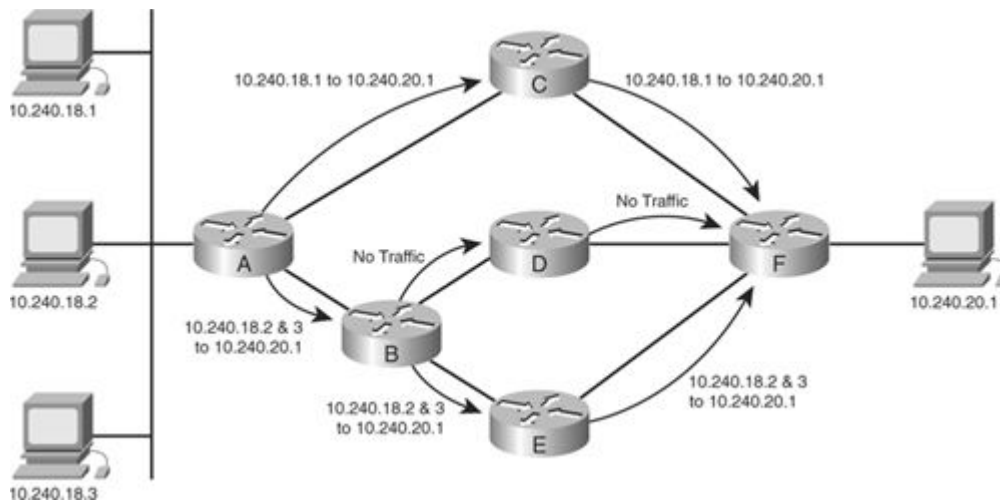
Hash	OK	Interface	Address	Packets
1	Y	Serial0/0	point2point	0
2	Y	Serial0/2	point2point	0
3	Y	Serial0/1	point2point	0
4	Y	Serial0/0	point2point	0
5	Y	Serial0/2	point2point	0
6	Y	Serial0/1	point2point	0
7	Y	Serial0/0	point2point	0
8	Y	Serial0/2	point2point	0
9	Y	Serial0/1	point2point	0
10	Y	Serial0/0	point2point	0
11	Y	Serial0/2	point2point	0
12	Y	Serial0/1	point2point	0
13	Y	Serial0/0	point2point	0
14	Y	Serial0/2	point2point	0
15	Y	Serial0/1	point2point	0

CEF Polarization and Load-Sharing Algorithms

One problem related to the CEF system of load sharing by hashing across the source and destination addresses is CEF polarization. CEF polarization occurs when all routers in a chain use the same hash algorithm to reach a destination and the traffic is polarized instead of load shared as desired. Figure 6-9 illustrates a situation where CEF polarization takes place.

Figure 6-9. CEF Load-Sharing Polarization

[View full size image]



Here is the packet flow through Figure 6-9:

Traffic sourced from 10.240.18.1 and destined to 10.240.20.1 enters the network at Router A and is CEF switched. Because there are two equal cost paths to the 10.240.20.0/24 network, the source and destination addresses in the packet go through the hash algorithm, and the result is a specific path used to reach the destination. In this case, the path the packets will take is toward Router C. From there, the packets go to Router F and on to their final destination.

Traffic sourced from 10.240.18.2 and destined to 10.240.20.1 enters the network at Router A and is CEF switched as well. Because there are two equal-cost paths to the 10.240.20.0/24 network, the source and destination addresses in the packet go through the hash algorithm and CEF chooses a path. In this case, the path the packets will take is toward Router B.

Traffic sourced from 10.240.18.3 and destined to 10.240.20.1 enters the network at Router A and is also CEF switched. Because there are two equal-cost paths to the 10.240.20.0/24 network, the source and destination addresses in the packet go through the hash algorithm and CEF chooses a path. In this case, the path the packets will take is toward Router B.

The packets sourced from 10.240.18.2 and 10.240.18.3 both arrive at Router B, which again has two equal-cost paths to reach 10.240.20.1. It will again run these sets of source and destination pairs through the hash algorithm, which will produce the same results as the hash algorithm on Router A produced. This means that both streams of packets pass along one path—in this case, the link toward Router E. The link toward Router D receives no traffic.

After the traffic sourced from 10.240.18.2 and 10.240.18.3 is received on Router E, it will be switched along the path to Router F and then on to its final destination.

The problem with the packet-switching decisions in this scenario is that both routers use the same hash algorithm, each with equal-cost paths toward the same destination. Using the same pair of source and destination addresses against the hash algorithm will return the same result. The router assigns traffic from the source and destination pair to one link, and the next router will run the same algorithm with the same result. Hence, traffic is not load shared as desired.

There are several possible solutions to this polarization. Per-packet load sharing is an option, but it is not an acceptable alternative because the network design will cause out-of-order packets.

The recommended solution, which the Cisco engineers adopted, was to seed the hash algorithm on each router with a unique identifier (ID) to eliminate the polarization effect. As each router boots, CEF bases a seed on a 32-bit number (usually the highest loopback IP address), so the algorithm will produce a different result on each router, even though it is fed the same source and destination pair of addresses.

Note

Cisco patented this solution to the problem of polarization in hash algorithms; therefore, specific details are not provided here.

The result was an additional CEF configuration command:

```
Router(config)#ip cef load-sharing algorithm ?
```

```
original Original algorithm
```

```
tunnel Algorithm for use in tunnel only environments
```

```
universal Algorithm for use in most environments
```

There are three CEF load-sharing algorithms:

Original

Universal

Tunnel

Table 6-7 lists the three algorithms and their characteristics.

Algorithm	Description	Polarization Fix	Tuned for Tunnels
Original	Unseeded hash algorithm	No	No
Universal	Seeded hash algorithm	Yes	No
Tunnel	Seeded hash algorithm	Yes	Yes

The original algorithm experienced issues with polarization with CEF per-destination load sharing. Hence, the universal algorithm addresses this issue. The tunnel algorithm has a different hash function and works better in a tunnel environment. The next sections describe each algorithm in more detail.

Original Algorithm

The CEF polarization example in Figure 6-9 describes the original algorithm operation. The original algorithm is unseeded and the only option available in early Cisco IOS versions. Table 6-7 shows that the original algorithm also is not tuned for load sharing across tunnels.

Universal Algorithm

The universal algorithm is the default in current Cisco IOS versions. However, not all platforms support the universal algorithm. Refer to Cisco.com or your Cisco account team for platform feature support information. You can manually choose the basis for the seed for the universal algorithm, although you don't need to:

```
Router(config)#ip cef load-sharing algorithm universal ?
```

```
<1-FFFFFFFF> Fixed ID
```

```
<cr>
```

Use the show ip cef summary command to verify the unique ID, as seen in Example 6-39.

Example 6-39. Verifying the Load-Sharing Algorithm with the show ip cef summary Command

```
R3#show ip cef summary
IP CEF with switching (Table Version 90), flags=0x0
 26 routes, 0 reresolve, 0 unresolved (0 old, 0 new), peak 6
 26 leaves, 21 nodes, 25240 bytes, 101 inserts, 75 invalidations
 1 load sharing elements, 328 bytes, 1 references
universal per-destination load sharing algorithm, id 05767213
 3 CEF resets, 5 revisions of existing leaves
Resolution Timer: Exponential (currently 1s, peak 1s)
 0 in-place/0 aborted modifications
refcounts: 1099 leaf, 1086 node
Adjacency Table has 1 adjacency
R3#
```

Tunnel Algorithm

Out of development of the universal algorithm came the tunnel algorithm. With per-destination load sharing and universal algorithm, equal distribution occurs with a large number of sessions. With a small number of sessions, unequal distribution will occur and create another type of polarization. In a tunnel scenario, a few

tunnels can carry transparently numerous sessions within the tunnel. However, the tunnels only represent a small number of sessions to the router.

The tunnel algorithm overcomes the unequal load distribution seen with a small number of sessions. Tunnel technologies such as Multiprotocol Label Switching (MPLS) Traffic Engineering, generic routing encapsulation (GRE), or L2TP can benefit from the tunnel algorithm. This algorithm uses the unique ID to prevent traffic polarization like the universal algorithm. Use the `ip cef load-sharing algorithm tunnel <id>` command to set this value.

All of these algorithms are available in software, depending on the IOS version. In recent IOS versions, the universal algorithm is the default. In hardware-forwarding platforms, support might or might not be present. For example, the Cisco 3550 switch and most GSR line cards do not support the tunnel algorithm in hardware.

Hardware Platform Implementations

Note that some of the CEF hardware implementations are not the same as the software implementations across platforms. For example, some of the hardware CEF implementations do not support the unique ID in the hash to avoid the polarization issue. For example, load sharing with a Cisco SUP2 occurs in the PFC2. CEF polarization is a known limitation of the PFC2 hardware because the SUP2 only supports the original CEF algorithm. A workaround is to change the hardware load-sharing algorithm from only using the default of source and destination pairs to also utilizing Layer 4 ports. This allows more diversity in the load sharing although it does not fix the polarization issue. On the SUP2, you can use Layer 4 information for load sharing by enabling the `mls ip cef load-sharing full` command in native mode or the `set mls cef load-balance full` command in hybrid mode.

The SUP720 has the universal ID implementation through the default universal algorithm, unlike the SUP2. The SUP720 can also load-share using the Layer 4 information by implementing the `mls ip cef load-sharing full` command. With native and hybrid mode on the SUP720, the `mls ip cef load-sharing` command controls load sharing in hardware. Unlike the SUP2, SUP720 configuration occurs on the route processor (RP) side in both modes. DDTS CSCef77386 documents this issue. If, however, you want to load-share with the full option, the universal ID is no longer part of the CEF hash algorithm, and the algorithm only uses the source-destination pairs and the Layer 4 ports.

There is also an `mls ip cef load-sharing simple` command option. The simple command option still uses the default universal algorithm, but it allows more uniform load sharing with an even number of multiple paths on the SUP720.

You can also run the `mls ip cef load-sharing full simple` command. This allows Layer 4 port utilization with simple mode instead of using the universal algorithm and only the source and destination pairs.

This chapter covers the following topics:

- An Internet service provider's simple MPLS VPN design
- Understanding the CEF and MPLS relationship
- CEF considerations when troubleshooting MPLS VPN across various platforms
- CEF and MPLS VPN load-sharing considerations

Understanding CEF operation in a Multiprotocol Label Switching (MPLS) environment is important for design stability and troubleshooting. This chapter analyzes CEF in MPLS Virtual Private Network (VPN) applications across Cisco platforms. Service providers and enterprises are increasingly using MPLS to provide connection-oriented services using an IP infrastructure. They also deploy MPLS VPNs for reasons including scalability, security, and flexibility. Because of increased MPLS VPN usage, you need to understand CEF's role, especially when troubleshooting.

CEF is the fundamental switching path for MPLS. Without CEF, MPLS forwarding does not occur. MPLS forwarding relies heavily on the IP routing table and the CEF architecture. Therefore, MPLS VPN relies on CEF because MPLS VPN depends on MPLS for successful operation.

This chapter assumes a basic understanding of MPLS VPNs. Although issues can occur with MPLS VPN setup, label distribution protocol (LDP), and operation, many times the issues relate to label mismatches stored in CEF or some other CEF inconsistency. Sometimes administrators believe the issue is with CEF, but it is with routing design. This chapter discusses some of these issues across various Cisco platforms.

An Internet Service Provider's Simple MPLS VPN Design

An ISP network typically consists of edge routers (often called provider edge, or PE, routers) that connect to customer edge (CE) routers and other service provider devices. In [Figure 7-1](#), the ISP's PE routers receive customer routes (VPNv4 routes) through external BGP (eBGP) or some other routing protocol and send them through internal BGP (iBGP) to all other PE routers. Most networks use route reflectors to avoid the full mesh of iBGP connections, so all iBGP speakers have a session to the route reflectors. This is the basic structure of an ISP network for MPLS VPN implementation. [Table 7-1](#) defines the type of devices in an MPLS VPN network and their awareness of MPLS VPN.

Figure 7-1. Typical ISP MPLS VPN Implementation
[\[View full size image\]](#)

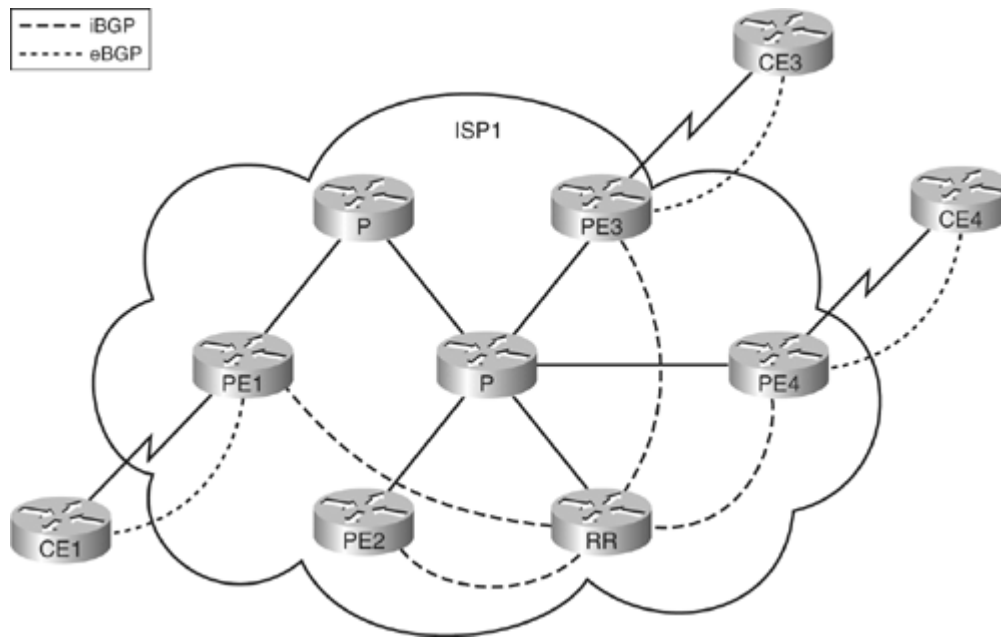


Table 7-1. MPLS VPN Device Type and Awareness

Type of Device	Description	MPLS VPN Awareness
Provider (P) router	Part of provider network that interfaces with PE and other P routers	No
Provider edge (PE) router	Part of the provider network that interfaces with the CE router and P and PE routers	Yes
Customer edge (CE) router	Part of the customer network that interfaces with the PE	No

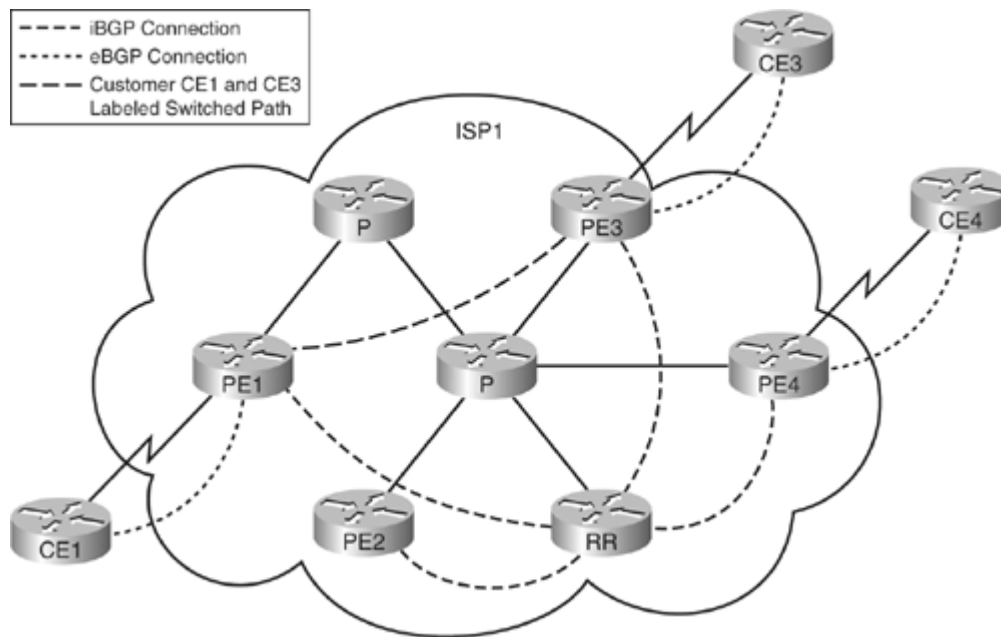
MPLS VPN uses a shared infrastructure but provides security for customers' traffic. If site Routers CE1 and CE3 belong to the same customer and VPN, their traffic is segmented from other customers' traffic and marked with a label. The PE routers maintain a separate routing table for each customer VPN called a virtual routing and forwarding (VRF) table.

While the VRF tables provide the isolation between customers, the data from these routing tables is exchanged between PE routers to enable data transfer between sites attached to different PE routers. The core routers in the provider network (P routers) that provide transport across the provider's backbone are not aware of customer routes.

The transit across the backbone is performed by using the labels that had been exchanged for routes in the global IP routing table. So, in the most common form, two labels are added to customer traffic. The core routers do not care about the customers' IP packet for forwarding, but see a labeled packet targeted toward an egress PE router. The egress PE router performs label switching on the second label (VPN label that was previously assigned), and either forwards the IP packet toward the CE router or performs another IP lookup in the VRF. The end-to-end

path between CE1 and CE3 is an MPLS Label Switch Path (LSP) tunnel. Figure 7-2 illustrates a typical LSP between two sites.

Figure 7-2. Typical LSP Between Two Sites
[View full size image]



Understanding the CEF and MPLS VPN Relationship

An MPLS VPN depends on CEF functionality. MPLS creates its own database for lookups called the Label Forwarding Information Base (LFIB), but it uses the CEF FIB as a source of this information. If CEF cannot resolve a route, the label cannot switch to a destination either. The label process path of an MPLS VPN packet between customer sites is label imposition, label swapping, and then label disposition.

Label imposition occurs when the PE router forwards based on the IP header and adds an MPLS label to the packet upon entering an MPLS network. In the direction of label imposition, the router switches packets based on a CEF table lookup to find the next hop and adds the appropriate label information stored in the FIB for the destination.

When a router performs label swapping in the core on an MPLS packet, the router does an MPLS table lookup. The router derives this MPLS table (LFIB) from information in the CEF table and the Label Information Base (LIB). Path changes that occur to reach a destination can result in changes in the FIB and LFIB. Multiple paths known through different routers can result in one label for each path. The router derives label load-sharing information from the CEF load-sharing information. For load sharing, the router looks inside the packet to determine whether it is an IP version 4 (IPv4) packet. If it is an IPv4 packet, CEF uses the IP addresses for hashing. For example, in a P-P scenario, the P router looks below the MPLS labels and inside the IP header to use the IP addresses to allocate the correct hash bucket for load sharing but not to make a forwarding decision. The P router makes the forwarding decision based on the topmost label.

Label disposition occurs when the PE router receives an MPLS packet, makes a forwarding decision based on the MPLS label, removes the label, and sends an IP packet. The PE router uses the LFIB for path determination for a packet in this direction.

Table 7-2 summarizes which structures to look at when troubleshooting. As stated previously, a special iBGP session facilitates the advertisement of VPNv4 prefixes and their labels between PE routers. At the advertising PE, BGP allocates labels for the VPN prefixes learned locally and installs them in the LFIB, which is the MPLS forwarding table. From a PE device perspective, locally learned prefixes are prefixes learned from the connected

customer edge device by the PE. At the receiving PE, if BGP accepts the remotely learned VPN prefixes with labels based on the route-target importation, BGP installs the VPN prefixes in the VRF FIB, which is the CEF table. From the PE perspective, remotely learned prefixes are prefixes learned from another PE through eBGP. An IP lookup occurs only once at the ingress PE. In short, if the router receives an MPLS packet, lookup occurs in the LFIB. If the router receives an IP packet, lookup occurs in the FIB.

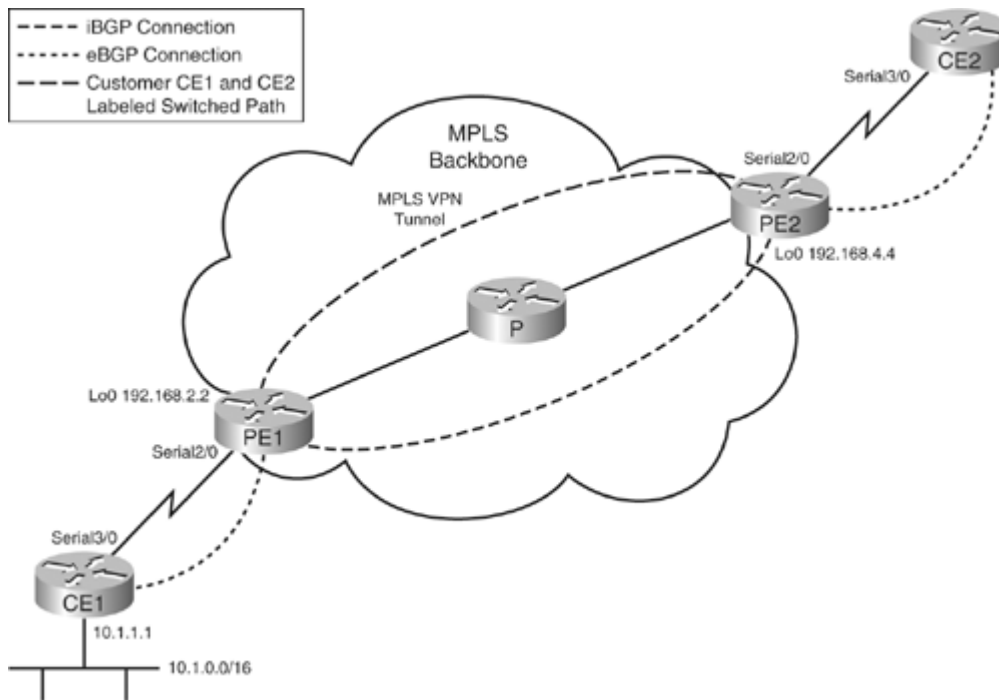
Table 7-2. Switching Structure Used by Function			
Structure	Command	Where to Use	Device Function
LFIB	show mpls forwarding_ <destination>	P routers	Label-to-label switching
LFIB	show mpls forwarding vrf <name> <destination> detail	Advertising PE/egress PE routers for locally learned prefixes	Label-to-IP switching (disposition)
VRF FIB	show ip cef vrf <name> <destination> detail	Receiving PE/ingress PE for remotely learned prefixes	IP-to-label switching (imposition)
FIB	show ip cef <destination>	CE	Normal routing/CEF switching

It is best to relate to an example to show how CEF is important in each stage of the label process path. The following cases show how to verify proper operation of an MPLS VPN network and how to analyze the LFIB and VRF CEF tables across an LSP.

Case 1: Label Disposition

In [Figure 7-3](#), an MPLS VPN tunnel (LSP) exists between PE1 and PE2. CE1 is advertising its LAN prefix 10.1.0.0/16 through eBGP to PE1. PE1 in turns advertises this local prefix through iBGP to other PE routers. PE1 must perform label disposition on traffic coming from the MPLS cloud toward 10.1.0.0/16. Therefore, to verify that PE1 has the information on how to reach 10.1.1.1 properly, you need to look at the LFIB on PE1. First, examine what label BGP assigns for this VPN prefix 10.1.0.0/16.

Figure 7-3. MPLS VPN Tunnel
[\[View full size image\]](#)



In [Example 7-1](#), PE1's BGP process sets the VPN label to 20.

Example 7-1. Verifying BGP Label Assignment on Egress PE for 10.1.0.0/16

```
PE1#show ip bgp vpn vrf red label |include 10.1.0.0
 10.1.0.0/16 10.3.1.2          20/nolabel
PE1#show ip bgp vpn vrf red 10.1.1.1
BGP routing table entry for 1::10.1.0.0/16, version 25550
Paths: (1 available, best #1, table red)
  Advertised to non peer-group peers:
    192.168.4.4
    65001
    10.3.1.2 from 10.3.1.2 (10.3.1.6)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: RT:1:1,
      mpls labels in/out 20/nolabel
```

Now, check the LFIB to make sure that PE1 sees the local, expected label to receive as 20. In [Example 7-2](#), notice that the LFIB sees the correct local label/tag as 20.

Example 7-2. Verifying the LFIB on Egress PE for 10.1.0.0/16

```
PE1#show mpls forwarding vrf red 10.1.1.1
Local   Outgoing   Prefix          Bytes tag   Outgoing     Next Hop
tag     tag or VC  or Tunnel Id   switched   interface
20      Untagged  10.1.0.0/16[V] 0           Se2/0       point2point
```

Note

The PE router typically installs a locally learned route from a CE in the VRF routing table, VRF CEF table, and LFIB. However, the PE usually only checks the LFIB when passing traffic to a CE because traffic destined for

the CE is typically an incoming MPLS packet. A PE router can also receive an IP packet destined for a CE if two CE routers are in the same VRF, connected to the same PE router, and sending traffic to each other. So, if an incoming packet to a PE is an IP packet from a VPN site, look in the VRF CEF table. If an incoming packet is an MPLS packet, look in the LFIB.

Case 2: Label Imposition

For CE2 to reach CE1, PE2 must perform label imposition on an IP packet received from CE2. At the other end of the VPN tunnel, PE2 should also see the correct VPN label of 20 to reach 10.1.1.1. Look at the BGP table and the VRF FIB to confirm this, as shown in [Example 7-3](#).

Example 7-3. Verifying BGP and VRF FIB on Ingress PE for 10.1.0.0/16

```
PE2#show ip bgp vpn vrf red 10.1.1.1
BGP routing table entry for 1:1:10.1.0.0/16, version 8
Paths: (1 available, best #1, table red)
  Advertised to non peer-group peers:
    10.4.1.2
    65001
      192.168.2.2 (metric 21) from 192.168.2.2 (192.168.2.2)
        Origin IGP, metric 0, localpref 100, valid, internal, best
        Extended Community: RT:1:1,
        mpls labels in/out nolabel/20
PE2#show ip cef vrf red 10.1.1.1
10.1.0.0/16, version 7, epoch 0, cached adjacency 172.16.3.1
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0, 172.16.3.1, tags imposed: {16 20}
  via 192.168.2.2, 0 dependencies, recursive
  next hop 172.16.3.1, Ethernet1/0 via 192.168.2.2/32
  valid cached adjacency
  tag rewrite with Et1/0, 172.16.3.1, tags imposed: {16 20}
```

In [Example 7-3](#), the out label on PE2 to reach 10.1.1.1 is 20. This is correct. Two labels are found in the FIB output. The second label/bottom label is the BGP label. The first label/top label is the IGP label. The core P devices use the IGP label to reach the egress PE. Therefore, the IGP label must also be correct to reach PE1 from PE2 across the MPLS cloud. The IGP label is 16 in this output. The next case verifies whether this IGP label is correct. PE2 uses PE1's loopback address to reach 10.1.1.1 because it is the BGP next hop, as seen in [Example 7-4](#).

Example 7-4. Verifying the VPN Path to Reach 10.1.1.1 from the Ingress PE

```
PE2#show ip route vrf red 10.1.1.1
Routing entry for 10.1.0.0/16
  Known via "bgp 65100", distance 200, metric 0
  Tag 65001, type internal
  Last update from 192.168.2.2 3d13h ago
  Routing Descriptor Blocks:
    * 192.168.2.2 (Default-IP-Routing-Table), from 192.1
      Route metric is 0, traffic share count is 1
      AS Hops 1
PE2#show ip route 192.168.2.2
Routing entry for 192.168.2.2/32
  Known via "ospf 100", distance 110, metric 21, type intra area
```

```

Last update from 172.16.3.1 on Ethernet1/0, 01:38:42 ago
Routing Descriptor Blocks:
* 172.16.3.1, from 192.168.2.2, 01:38:42 ago, via Ethernet1/0
  Route metric is 21, traffic share count is 1

```

Case 3: Label Swapping

On the P router, the local tag to reach 192.168.2.2, PE1's loopback, is 16, as seen in [Example 7-5](#). Therefore, for traffic destined for 192.168.2.2, the P router expects an incoming label of 16.

Example 7-5. Verifying the LFIB on the P Router

```

P#show mpls forwarding 192.168.2.2
Local   Outgoing   Prefix          Bytes tag   Outgoing     Next Hop
tag     tag or VC  or Tunnel Id   switched   interface
16      Pop tag    192.168.2.2/32 4350468    Et0/0        172.16.2.1

```

Therefore, the IGP label that PE2 sees is correct in [Example 7-3](#). When the P router receives traffic toward 10.1.1.1 with a label of 16, it pops the top label and the packet passes to PE1 with only the bottom label of 20, the VPN label.

Troubleshooting an MPLS VPN

The previous section outlined the proper operation of an MPLS VPN network and basic verification of the LFIB and VRF CEF tables. However, ambiguity can exist when traffic is not passing properly when it was previously. A PE might not have installed the BGP label into the CEF FIB. If so, clear the VRF route to resolve the issue and look for a software bug using the Cisco Bug Toolkit. You can enable the debug mpls lfib cef command to get more information about LFIB additions and deletions. This command can be useful in troubleshooting label mismatches in FIB and LFIB but can be intensive on production routers. Therefore, use it with care. [Table 7-3](#) lists some key tips and commands when troubleshooting an MPLS VPN problem.

Table 7-3. Typical Problems Seen When Troubleshooting an MPLS VPN

Symptom	Tip	Command
VPNv4 traffic is not getting forwarded end to end	Check the label information in BGP and LFIB at the advertising PE router.	show ip bgp vpn vrf <vrf> label include <prefix> show mpls forwarding vrf <vrf> include <prefix>
	Check the label information in BGP and FIB at the receiving PE router.	show ip bgp vpn vrf <vrf> label include <prefix> show ip cef vrf <vrf> <prefix>
Incoming MPLS traffic is dropped at the egress PE	Check the FIB and LFIB entries on the route processor (RP), line cards (LCs), and relevant hardware engines if present.	show mpls forwarding vrf <prefix>

Table 7-4 lists helpful show commands.

Table 7-4. Helpful show Commands When Troubleshooting an MPLS VPN	
show Command	Purpose
show ip route vrf <vrf>	Shows the route entries relevant to the specified VRF only
show ip arp vrf <vrf>	Shows Address Resolution Protocol (ARP) entries relevant to the specified VRF only
show ip cef vrf <vrf> <prefix>	Shows the label stack and outgoing interface for the particular prefix
show mpls forwarding vrf <vrf> <prefix>	Shows the LFIB lookup based on a VPN prefix
show mpls forwarding include <prefix>	Shows whether a prefix is in the LFIB
show mpls forwarding label <label>	Shows the LFIB lookup based on the incoming label
show ip cef vrf <name> exact-route <source address> <destination address>	Shows the path of traffic from the PE's perspective for a VPN prefix

Table 7-5 lists helpful debug commands.

Table 7-5. Helpful debug Commands When Troubleshooting MPLS VPNs	
debug Command	Purpose
debug ip bgp vpnv4	Useful when troubleshooting label-related problems in BGP
debug mpls lfib cef [acl]	Useful when troubleshooting label mismatch among BGP, FIB, and LFIB for VPN prefixes
debug ip routing vrf <vrf> [acl]	Useful when router does not install VPN prefixes in the VRF routing table

CEF Considerations When Troubleshooting MPLS VPN Across Various Platforms

When troubleshooting MPLS VPNs on Cisco devices, you need to understand the platform architecture. Different architectures and processors require different commands to troubleshoot. Troubleshooting is different

on a software-based platform such as a Cisco 7200 router with an NPE-G2 processor. More troubleshooting steps must be taken on a distributed platform such as the Cisco 7500 to check the line card information. With the Cisco Catalyst 6500, the type of supervisor determines the device usage possible in an MPLS VPN environment and the type of troubleshooting. Hardware-based forwarding on a Cisco 12000 series router varies with line card engine type. With advanced technology, Parallel Express Forwarding (PXF) requires additional troubleshooting on a Cisco 10000 series router. Caveats of these platforms are discussed in the following sections.

Cisco 7200 Router with an NPE-G2

Many customers use the Cisco 7200 with an NPE-G1 or NPE-G2 in MPLS environments. On these platforms serving as a PE device, the same rules previously covered apply. The NPE-G2 has optimized performance but does not do hardware-based switching. So, when troubleshooting an MPLS VPN issue, you have to examine the routing, CEF, and LFIB tables in one place only. This is also true for other software-based platforms such as the Cisco ISR routers.

Cisco 7500 Router

On the other hand, on a distributed platform such as a Cisco 7500 router acting as a PE device, you need to check the routing, CEF, and LFIB tables on the route processor level and on the line card level. On a 7500 with Versatile Interface Processors (VIPs), most of the time forwarding occurs on the VIP. Sometimes inconsistencies exist between the LFIB on the route switch processor (RSP) and on the VIP. This should not be the case if forwarding is happening on the VIP. If the LFIB and other information look fine on the RSP but traffic forwarding issues remain, you might have to look at the LFIB on the VIP. If an inconsistency exists, it most likely is a software bug. After attaching to a VIP, you can run the same commands that are run on an RSP to troubleshoot MPLS issues and check consistency with commands such as `show mpls forwarding <prefix>` or `show ip cef <prefix>`. [Example 7-6](#) shows that the labels and next hops are consistent to reach 10.1.1.1/32 on the RSP and VIP.

Example 7-6. Checking Consistency for Prefix 10.1.1.1/32 on RSP and VIP

```

7513#show mpls forwarding 10.1.1.1
Local  Outgoing  Prefix          Bytes tag  Outgoing     Next Hop
tag   tag or VC  or Tunnel Id   switched  interface
17    Untagged  10.1.1.1/32   0         Et9/1/0      10.1.16.1
7513#if-con 9
Console or Debug [C]:
Entering CONSOLE for VIP2 R5K 9
Type "^C^C^C" or "if-quit" to end this session

VIP-Slot9#show mpls forwarding 10.1.1.1
Local  Outgoing  Prefix          Bytes tag  Outgoing     Next Hop
tag   tag or VC  or Tunnel Id   switched  interface
17    Untagged  10.1.1.1/32   0         Et9/1/0      10.1.16.1

```

In one case, an MPLS inconsistency between the VIPs and RSP is not a bug. By default, slow serial interfaces (less than 2 Mbps) use RSP-based weighted fair queuing (WFQ). If an interface has RSP WFQ enabled, the VIP punts a packet destined for the VRF prefixes out these interfaces to the RSP, and then the RSP performs WFQ and switches the packets. A distributed CEF punt happens and the incoming VIP does not store the incoming label/LFIB information because of RSP-based WFQ. It is desirable for the VIP rather than the RSP to forward the traffic. Therefore, you should enable VIP-based fair queuing instead of RSP-based fair queuing. With VIP-based fair queuing enabled, label switching and WFQ will occur on the VIP.

Cisco Catalyst 6500 with a Supervisor 2

On a Cisco Catalyst 6500 with a Supervisor 2 (SUP2), MPLS VPN troubleshooting is different than on routers because the architecture is different. In general, the ingress and egress interfaces must be FlexWAN or Optical Services Module (OSM) PXF-enabled interfaces to support MPLS VPNs. This is because these cards are MPLS aware and required for imposition and disposition. Because the Policy Feature Card 2 (PFC2) switching ASICs

cannot handle MPLS packets, MPLS tables are loaded directly to OSM or FlexWAN cards from the route processor to handle switching.

The general troubleshooting at the software level remains the same as that discussed in the section "Troubleshooting an MPLS VPN," earlier in this chapter. However, on a Cisco Catalyst 6500 platform, the main benefit is hardware switching. This factor adds another twist to troubleshooting MPLS VPNs. The switch does not use the Supervisor 2's FIB Ternary Content Addressable Memory (TCAM) for MPLS functionality. The OSM or FlexWAN stores all the MPLS and VPN forwarding tables. The route processor passes the LFIB table to the Toaster Tag FIB (TTFIB) on each card. The TTFIB is a condensed form of the LFIB for optimized use with the PXF processor and FlexWAN. The OSM and FlexWAN line cards perform the actual lookup and rewrite for MPLS packets in hardware using the TTFIB.

For a Cisco Catalyst 6500SUP2 serving as an MPLS PE device, the ingress card determines the VPN that the IP packet belongs to and performs a lookup in the TTFIB to determine the egress interface. The ingress index directs the IP packet to the egress port, bypassing the PFC2. The source MAC address is encoded with a TTFIB index so that the egress is able to determine the proper lookup and labels. The egress adds the two labels—one VPN and one IGP label—and sends the packet out.

In the reverse direction of label disposition, an MPLS packet arrives and the SUP2 uses the label to determine the VPN owner for the packet. The ingress index directs the packet to the egress interface with special information. The egress pops the label, does a TTFIB lookup, and forwards the IP packet.

When a Cisco Catalyst 6500 SUP2 device serves as a P device, you must have a FlexWAN or OSM as the ingress and egress card. The ingress card does a lookup in the TTFIB based on the label to determine the egress interface. The device forwards the packet directly from the ingress to the egress. The packet arrives at the egress module and it does another TTFIB lookup. Then the egress performs the appropriate label swap.

When troubleshooting MPLS VPNs on a SUP2 platform with OSMs or FlexWANs, examination must occur on the hardware tables of these cards. Therefore, in troubleshooting, you must compare the output of the MPLS forwarding table on the RP with the TTFIB on the ingress line card. On the line card, check the FIB, LFIB, and hardware CEF entries.

Catalyst 6500 with a Supervisor 720 3BXL

On a Supervisor 720 3BXL, the EARL 7/PFC3 switching ASICs can handle MPLS packets and there is no longer a requirement to have an OSM or FlexWAN card as the ingress/egress for imposition and disposition. The Layer 3 ASIC on the EARL 7 populates the FIB TCAM with MPLS entries. It also handles label imposition, swapping, and disposition. With a SUP720, any line card is capable of being an MPLS link. The PFC3/DFC3 perform the MPLS VPN forwarding decision for OSM PXF-enabled interfaces and FlexWAN interfaces unlike in a PFC2 system, where the OSM PXF interfaces and FlexWAN perform the forwarding decision. The RP on the SUP720 builds the VRF entries. It sends information to the SP to program the TCAM. The FIB TCAM stores a VPN table ID along with the label and prefix entry. [Table 7-6](#) shows helpful commands to use when troubleshooting a P device that has an SUP720 in an MPLS environment.

Table 7-6. Helpful P Troubleshooting Commands on Native SUP720s

show Command	Function
show mpls forwarding <prefix> detail	Shows software information
show mls cef <prefix> detail	Shows hardware forwarding information
show mls cef adj entry <index> detail	Shows hardware adjacency

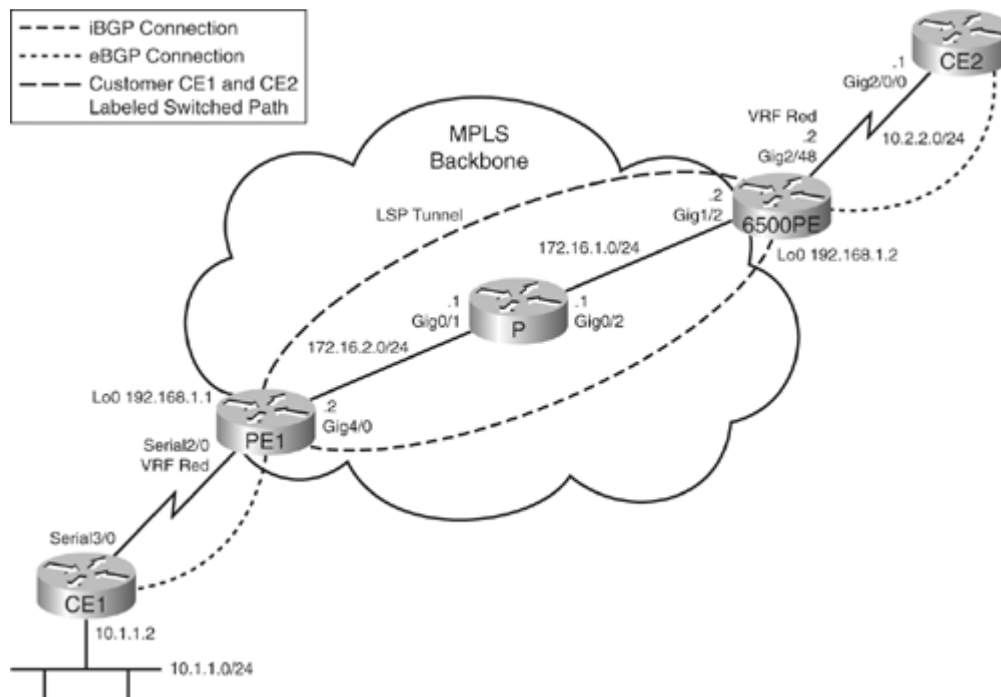
Table 7-7 shows helpful commands to use when troubleshooting an MPLS VPN PE device that is a Cisco Catalyst 6500 with a SUP720.

Table 7-7. Helpful PE Troubleshooting Commands on Native SUP720s	
show Command	Function
show ip route vrf <name> <prefix>	Verifies routing entry
show ip cef vrf <name> <prefix> detail	Verifies CEF entry
show mpls forwarding vrf <name> <prefix> detail	Shows software forwarding information
show mls cef vrf <name> <prefix> <mask length> detail	Shows hardware forwarding information
show mls cef adjacency entry <index> detail	Shows hardware adjacency information
show vlan internal usage	Shows internal VLAN mapping and outgoing interface
show mls cef mpls labels <label> detail	Shows hardware label information

Figure 7-4 shows a Cisco Catalyst 6500 with a SUP720 used as an MPLS VPN PE device.

Figure 7-4. SUP720 Used as an MPLS VPN PE

[\[View full size image\]](#)



Example 7-7 includes show commands for checking software data for VRF prefix 10.1.1.0/24 on a SUP720 serving as a PE device in an MPLS VPN network. In this example, the VRF name is "red."

Example 7-7. Checking Software Information for Prefix 10.1.1.0/24 in VRF Red

Code View: [Scroll](#) / [Show All](#)

```
6500PE#show ip route vrf red 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "bgp 65001", distance 200, metric 0, type internal
  Last update from 192.168.1.1 00:05:07 ago
  Routing Descriptor Blocks:
  * 192.168.1.1 (Default-IP-Routing-Table), from 192.168.1.1, 00:05:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 0
6500PE#show ip cef vrf red 10.1.1.0 detail
10.1.1.0/24, version 33, epoch 0, cached adjacency 172.16.1.1
0 packets, 0 bytes
  tag information set, all rewrites owned
  local tag: VPN-route-head
  fast tag rewrite with Gi1/2, 172.16.1.1, tags imposed: {17 22}
  via 192.168.1.1, 0 dependencies, recursive
  next hop 172.16.1.1, GigabitEthernet1/2 via 192.168.1.1/32 (Default)
  valid cached adjacency
  tag rewrite with Gi1/2, 172.16.1.1, tags imposed: {17 22}
6500PE#show mpls forwarding vrf red 10.1.1.0 detail
Local   Outgoing   Prefix           Bytes tag   Outgoing   Next Hop
tag     tag or VC   or Tunnel Id     switched   interface
None    17          10.1.1.0/24      0          Gi1/2      172.16.1.1
        MAC/Encaps=14/22, MRU=1496, Tag Stack{17 22}
        000F35E4541900E0142B63008847 0001100000016000
        No output feature configured
        Per-packet load-sharing
```

Example 7-8 checks the hardware information on the SUP720 for the prefix 10.1.1.0/24 in VRF red. To view the hardware adjacency entry, you must identify the index from the first command's "A:" value.

Example 7-8. Checking Hardware Information for Prefix 10.1.1.0/24 in VRF Red

Code View: [Scroll](#) / [Show All](#)

```
6500PE#show mls cef vrf red 10.1.1.0 24 detail
Codes: M - mask entry, V - value entry, A - adjacency index, P - priority
bit
      D - full don't switch, m - load balancing modnumber, B - BGP Bucket
sel
      V0 - Vlan 0, C0 - don't comp bit 0, V1 - Vlan 1, C1 - don't comp bit 1
      RVTEN - RPF Vlan table enable, RVTSEL - RPF Vlan table select
Format: IPV4_DA - (8 | xtag vpn pi cr recirc tos prefix)
Format: IPV4_SA - (9 | xtag vpn pi cr recirc prefix)
M(3211  ): E |1 FFF  0 0 0 0  255.255.255.0
V(3211  ): 8 |1 256  0 0 0 0  10.1.1.0          (A:294912, P:1,D:0,m:0
,B:0
)
6500PE#show mls cef adjacency entry 294912 detail
Index: 294912 smac: 00e0.142b.6300, dmac: 000f.35e4.5419
          mtu: 1518, vlan: 1020, dindex: 0x0, l3rw_vld: 1
```

```
format: MPLS, flags: 0x8418
label0: 0, exp: 0, ovr: 0
label1: 22, exp: 0, ovr: 0
label2: 17, exp: 0, ovr: 0
op: PUSH_LABEL2_LABEL1
packets: 0, bytes: 0
```

In the hardware adjacency, label2 has a value of 17 and label1 has a value of 22. Traffic destined for 10.1.1.0 in VRF red will have both labels attached. This matches the data for CEF and the LFIB seen in [Example 7-7](#). Looking at the internal VLAN usage for the device can give the outbound interface mapping. In this case, VLAN 1020 maps to interface GigabitEthernet1/2, as seen in [Example 7-9](#).

Example 7-9. Verifying Internal VLAN Mapping and Outbound Interface

```
6500PE#show vlan internal usage

VLAN Usage
-----
1006 online diag vlan0
1007 online diag vlan1
1008 online diag vlan2
1009 online diag vlan3
1010 online diag vlan4
1011 online diag vlan5
1012 PM vlan process (trunk tagging)
1013 Control Plane Protection
1014 L3 multicast partial shortcuts for VPN 0
1015 Egress internal vlan
1016 Multicast VPN 0 QOS vlan
1017 IPv6 Multicast Egress multicast
1018 vrf_0_vlan0
1019 GigabitEthernet2/48
1020 GigabitEthernet1/2
```

When troubleshooting a Cisco Catalyst 6500 platform, you should understand proper operations, such as the path of the packet and the expected label value. You should find consistency among the routing table, MPLS forwarding table, and the LFIB.

Cisco 12000 Series Router

On a Cisco 12000, forwarding occurs through distributed CEF switching. Some of the line cards are also capable of hardware-based forwarding, unlike a VIP. In this case, not only should you examine the route processor's and line cards' software FIB and LFIB tables, but you must also examine the hardware table on the engine cards because that is where forwarding occurs. The commands used depend on the type of engine cards and the router's function.

Cisco 12000 line cards, which have an Engine 0 or an Engine 1, use software forwarding on the line card for MPLS packets. Engine 2 line cards have a packet-switching ASIC (PSA) that forwards MPLS packets in hardware. Engine 3 line cards use the alpha ASIC for packet forwarding in hardware. Engine 4 line cards use the gen6 ASIC and Engine 4+ cards use the gen7 ASIC. To troubleshoot MPLS or IP forwarding early in product life, you had to execute ASIC-based commands on the respective line cards. Because the ASICs used across various engines differed, it became confusing when troubleshooting. Hence, Cisco implemented standard commands for troubleshooting, as listed in [Table 7-8](#).

Table 7-8. Old Versus Standard Commands		
Engine	Old ASIC-Based or Tag Command	Standardized Command
2	show ip psa-cef	show ip hardware-cef
	show tag psa-tag show tag hardware-tag	show mpls hardware-lfib
	show ip psa-cef vrf	show ip hardware-cef vrf
3	show ip alpha-cef	show ip hardware-cef
	show tag alpha-tag show tag hardware-tag	show mpls hardware-lfib
4	show ip gen6-cef	show ip hardware-cef
4+	show ip gen7-cef	show ip hardware-cef
	show tag hardware-tag	show ip hardware-lfib

Table 7-9 shows the type of forwarding on each line card engine and some helpful troubleshooting commands on a PE router in an MPLS VPN environment.

Table 7-9. Helpful Commands to Troubleshoot MPLS VPNs on a PE		
Engine Type	Forwarding Type	Helpful Commands on Ingress Line Card
0	Software	show ip cef vrf <name> <prefix>
Engine 1	Software	Not applicable. Engine 1 does not have PE functionality in an MPLS VPN environment.
Engine 2	Hardware	show ip hardware-cef vrf <name> <destination prefix> <mask> detail show ip hardware-cef vrf <name> exact-route <source> <destination>
Engine 3	Hardware	show ip hardware-cef tofab vrf <name> <prefix> detail

Table 7-9. Helpful Commands to Troubleshoot MPLS VPNs on a PE

Engine Type	Forwarding Type	Helpful Commands on Ingress Line Card
		show ip hardware-cef tofab vrf <name> exact-route <source> <destination>
Engine 4	Hardware	Not applicable. Engine 4 does not have PE functionality in an MPLS VPN environment. Therefore, no VRF commands are available for hardware. The router prints an error message when the user configures ip vrf forwarding on a port on an Engine 4 card.
Engine 4+	Hardware	show ip gen7-cef vrf <name> <prefix> show ip gen7-cef vrf <name> exact-route <source> <destination> Note: At the time of this writing, the show ip hardware-cef command does not include VRF commands. Therefore, you must use the show ip gen7-cef command for troubleshooting.

Example 7-10 shows the output of troubleshooting VRFs on an Engine 3 line card. A key component is the output interface.

Example 7-10. Engine 3 VRF Verification

Code View: [Scroll](#) / [Show All](#)

```

gsr#execute-on slot 3 show ip hardware-cef tofab vrf test 10.99.99.99
detail
===== Line Card (Slot 3) =====

Root: 0x2419C000 Location: 0x2419C18C Data: 0x000717E0
Node: 0x2438BF00 Location: 0x2438BF60 Data: 0x00000000 0x000717C0
Node: 0x2438BE00 Location: 0x2438BE30 Data: 0x00000000 0x000717A0
Node: 0x2438BD00 Location: 0x2438BD60 Data: 0x00000000 0x00071780
Node: 0x2438BC00 Location: 0x2438BC30 Data: 0x00000000 0x00071760
Node: 0x2438BB00 Location: 0x2438BB60 Data: 0x00000000 0x00071740
Node: 0x2438BA00 Location: 0x2438BA30 Data: 0xA0000000 0x0400F0FC
Node for 10.99.99.99: 0x2438BA30

Leaf FCR 4 0x2438BA30 found 7 deep
Fast Adjacency:
alpha adjacency: 0x201E1F80
[0-7] ui 16 ai 380 oq 4080 in 2B ab 10 hl 18 gp 7 tl 0 loq 8CC0 3/3 mtu
1500
Output interface is GigabitEthernet3/3.13
current counters 0, 0 last reported 0, 0

Output Queue / Local Output Queue Bundle:
[0-7] output queue 0x4080 local output queue 0x8CC0
PLU leaf data: 0xA0000000 0x0400F0FC 0xA1020380 0x24000000
Mask bits: 0 Origin AS: 0 Source lookup drop: no
QOS group: 0 Traffic index: 0 Precedence not set
Default Route: no PBR enabled: no

```

```

gsr#execute-on slot 3 show ip hardware-cef tofab vrf test exact-route
10.1.1.1
10.99.99.99
===== Line Card (Slot 3) =====
Leaf FCR 4 0x2438BA30 found 7 deep
Fast Adjacency:
alpha adjacency: 0x201E1F80
[0-7] ui 16 ai 380 oq 4080 in 2B ab 10 hl 18 gp 7 tl 0 loq 8CC0 3/3 mtu
1500
packets 0 bytes 0

Output Queue / Local Output Queue Bundle:
[0-7] output queue 0x4080 local output queue 0x8CC0
10.1.1.1 -> 10.99.99.99 Interface: GigabitEthernet3/3.13

```

Table 7-10 lists some helpful troubleshooting commands when an ingress line card is functioning as a P router performing label switching.

Table 7-10. Helpful Commands to Troubleshoot MPLS VPNs on a P Router

Incoming Line Card Engine Type	Software-Based show Commands	Hardware-Based show Commands
0	show mpls forwarding	N/A
1		
2	show mpls forwarding	show ip hardware-cef <prefix> detail
3		
4		show mpls hardware-Ifib labels <low label> detail
4+		

Note

In earlier Cisco IOS Software codes, the show mpls hardware-Ifib command might not be available. However, the show tag hardware-tag command gives the same information.

Example 7-11 shows output for an Engine 3 serving as a P interface. The important data is the output interface and the tag information.

Example 7-11. Engine 3 LFIB Hardware Output

Code View: [Scroll](#) / [Show All](#)

```
prp-12008#execute-on slot 0 show ip hardware-cef 10.0.0.99 detail
===== Line Card (Slot 0) =====

Root: 0x240CE000 Location: 0x240D0800 Data: 0x00898000
Node: 0x284C0000 Location: 0x284C0000 Data: 0x00000000 0x01017C00
Node: 0x2C0BE000 Location: 0x2C0BE630 Data: 0xA0000000 0x06004316
Node for 10.0.0.99: 0x2C0BE630

Leaf FCR 6 0x2C0BE630 found 3 deep
Fast Tag Rewrite:
  [0-7]: ui 0 ai 7 oq 4080 in 11 ab 50 hl 20 gp 19 tl 4 loq 8000 0/0 mtu
4466
Output interface is POS0/0:1
      1 tag: 21
      current counters 0, 0 last reported 0, 0
Output Queue / Local Output Queue Bundle:
[0-7]  output queue 0x4080 local output queue 0x8000
```

Example 7-12 shows hardware output for an Engine 4+ serving as a P interface. The key components are the label values and the operation value. In this case, the operation is a push of label/tag 0 out slot 2 or slot 1. This matches the data in the LFIB.

Example 7-12. Engine 4+ LFIB and Hardware Output Comparison

Code View: [Scroll](#) / [Show All](#)

```
gsr#show mpls forwarding 10.0.0.99
Local   Outgoing   Prefix          Bytes tag   Outgoing   Next Hop
tag     tag or VC     or Tunnel Id    switched   interface
19      0             10.0.0.99/32    0          PO2/0      point2point
        0             10.0.0.99/32    0          PO1/0      point2point

gsr#execute-on slot 1 show ip hardware-cef 10.0.0.99 detail
===== Line Card (Slot 1) =====
HW Node : 70085000 = 9930300000203030
HW Node : 71018180 = E230200000443890
HW Node : 7221C480 = FF30100000506020
HW Node : 72830130 = 4A30000000601D40
HW Leaf :7300EA18, IP 10.0.0.99, Leaf: 6BB1670000602998: RPF 1, BGP 3 :Pkt
0, Byt 0
type=load bal: n=8 (addr=0x7300EA18, val=0x6BB1670000602998)
Hash 0 2 4 6 : TagLB ptrA entry=84B1600000140010
type=load bal: n=1 (addr=0x73014CC0, val=0x84B1600000140010)
0 TagLB ptrA entry=61B16000083000E7
type=load bal: n=1 (addr=0x70A00080, val=0x61B16000083000E7)
: TagLB ptrA entry=0, ptrB entry=75A1000000000000
type=mpls C: label #=1, tag =0, next=0x0 (addr=0x73800738,
val=0x75A1000000000000)
type=mpls A: ttl=0, cos=0, op=push, idx=0, adj=0xFFA00 ctr=0x3207B
(addr=0x71800738,
val=0x9A8040FFA003207B)
Lbl_L1 0x797FD000, Lbl Opcode 0, Is_Fast 1, Lbl_L2 ptr 0x797FD008
Lbl_L2[0] 797FD008, L2 opcode 0, OIF 0, LQF 0, OQF 0, S 0, Adj_L3
0x79001000, Bundle
```

```
1 Payload Type 0
AdjL3[0] 0x79001000, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000800 = 0, Byt 0
AdjL3[1] 0x79001010, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000808 = 0, Byt 0
AdjL3[2] 0x79001020, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000810 = 0, Byt 0
AdjL3[3] 0x79001030, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000818 = 0, Byt 0
AdjL3[4] 0x79001040, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000820 = 0, Byt 0
AdjL3[5] 0x79001050, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000828 = 0, Byt 0
AdjL3[6] 0x79001060, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000830 = 0, Byt 0
AdjL3[7] 0x79001070, MTU 0, Slot 2, LQ 0, OQ 4000, BHdr 1, OI 0x90100000,
Pkt
7A000838 = 0, Byt 0
Hash 1 3 5 7 : TagLB ptrA entry=7B1600000140011
type=load bal: n=1 (addr=0x73014CC8, val=0x07B1600000140011)
0 TagLB ptrA entry=2CB1600008300172
type=load bal: n=1 (addr=0x70A00088, val=0x2CB1600008300172)
: TagLB ptrA entry=0, ptrB entry=75A1000000000000
type=mpls C: label #=1, tag =0, next=0x0 (addr=0x73800B90,
val=0x75A1000000000000)
type=mpls A: ttl=0, cos=0, op=push, idx=0, adj=0xFF9BE ctr=0x32029
(addr=0x71800B90,
val=0x238040FF9BE32029)
Lbl_L1 0x797FCDF0, Lbl Opcode 0, Is_Fast 1, Lbl_L2 ptr 0x797FCDF8
Lbl_L2[0] 797FCDF8, L2 opcode 0, OIF 0, LQF 0, OQF 0, S 0, Adj_L3
0x79003D80, Bundle
1 Payload Type 0
AdjL3[0] 0x79003D80, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001EC0 = 0, Byt 0
AdjL3[1] 0x79003D90, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001EC8 = 0, Byt 0
AdjL3[2] 0x79003DA0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001ED0 = 0, Byt 0
AdjL3[3] 0x79003DB0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001ED8 = 0, Byt 0
AdjL3[4] 0x79003DC0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001EE0 = 0, Byt 0
AdjL3[5] 0x79003DD0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001EE8 = 0, Byt 0
AdjL3[6] 0x79003DE0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,
Pkt
7A001EF0 = 0, Byt 0
```



```
AdjL3[7] 0x79003DF0, MTU 0, Slot 1, LQ 0, OQ 4000, BHdr 5, OI 0x90100000,  
Pkt  
7A001EF8 = 0, Byt 0
```

Some important caveats exist for the Engine 4 and Engine 4+ line cards. The Engine 4 line card does not support MPLS load sharing in label-to-label or label-to-IP paths. If multiple paths exist for label-switching or label-disposition paths, the Engine 4 line card just chooses one to forward traffic if multiple paths exist. DDTS CSCdy41261 documents this issue and is viewable on the Cisco Bug Toolkit. This caveat does not affect the Engine 4+ line card.

Only Engines 0, 2, 3, and 4+ support MPLS VPN PE functionality totally with VPN imposition and VPN disposition.

Cisco 10000 Series Router

A Cisco 10000 is a PXF-based platform. If you have an MPLS issue, not only must you check the FIB/LFIB but you must also evaluate the PXF table, because that is where forwarding stems from. Some other PXF-based platforms do not support MPLS. In this case, these platforms punt the MPLS packets to the CEF path. Again, the key is to check that the labels and outgoing interfaces programmed in hardware are correct and consistent with the LFIB or FIB information, depending on the router's MPLS function.

Some helpful commands to troubleshoot a Cisco 10000 router include the following:

```
show hardware pxf cpu mpls labels <local label>  
show hardware pxf cpu cef <prefix>  
show hardware pxf cpu cef vrf <name> <prefix>
```

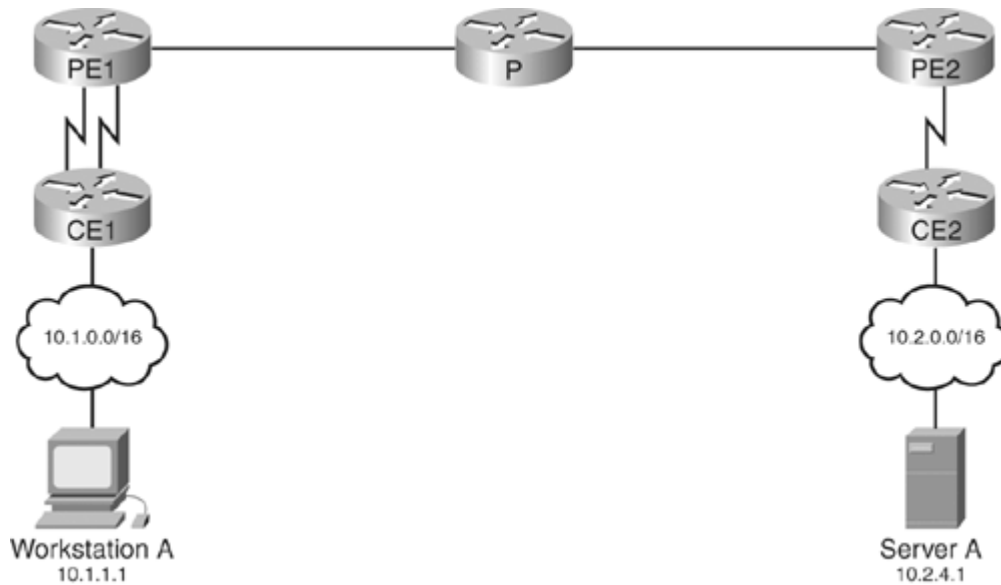
CEF and MPLS VPN Load-Sharing Considerations

Chapter 6, "Load Sharing with CEF," covers CEF and load sharing in an IP-only environment. The following sections address some special cases and troubleshooting techniques in an MPLS VPN environment.

PE-CE Load Sharing: CE Multihomed to Same PE

In **Figure 7-5**, Router CE1 is multihomed to Router PE1, and two paths exist between Routers CE1 and PE1. Many customers want to load-share across multiple connections between the same PE and CE routers. In the example shown, the number of paths used in the CE1-to-PE1 direction would depend on the routing table on CE1. The same is true in the PE1-to-CE1 direction, although PE1 uses the LFIB to reach CE1. Sometimes the use of MPLS or MPLS VPNs by the service providers confuses the customer. However, the principles remain the same with and without MPLS in this case. The CE has no knowledge of MPLS. Therefore, the load-sharing principles covered in **Chapter 6** all apply. The same is true with the PE, except that it uses the LFIB to pass traffic from the MPLS cloud to the CE.

Figure 7-5. PE-CE Load Sharing
[\[View full size image\]](#)



Consider a case where the PE and CE routers use external BGP to peer to each other's physical address. Note that the example shows a scenario that would require multipath BGP with or without the use of MPLS. CE1 and PE1 are running eBGP to share routes. [Example 7-13](#) looks at the routing table on CE1 to reach the server 10.2.4.1. Although two paths exist to reach 10.2.4.1, only one path is in the routing table. As seen in [Chapter 6](#), CEF will follow the routing table and only install one path in the FIB because that is all that is in the routing table.

Example 7-13. CE1's Routing and FIB Entries for 10.2.4.1

```
CE1#show ip route 10.2.4.1
Routing entry for 10.2.0.0/16
  Known via "bgp 65001", distance 20, metric 0
  Tag 65100, type external
  Last update from 10.3.1.1 00:04:40 ago
  Routing Descriptor Blocks:
  * 10.3.1.1, from 10.3.1.1, 00:04:40 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
CE1#show ip cef 10.2.4.1
10.2.0.0/16, version 14, epoch 0, cached adjacency to Serial3/0
0 packets, 0 bytes
  via 10.3.1.1, 0 dependencies, recursive
  next hop 10.3.1.1, Serial3/0 via 10.3.1.0/30
  valid cached adjacency
```

The issue is that BGP only passes the best path to the routing table. In [Example 7-14](#), the path through 10.3.1.1 is the best one. PE1 passes down both paths through the two BGP neighbor connections. However, BGP on the CE picks only one path as the best path. CEF is only going to use what is specified in the routing table. Therefore, the question is how to get BGP to install both paths in the CE's routing table.

Example 7-14. CE1's BGP Table

```
CE1#show ip bgp 10.2.4.1
BGP routing table entry for 10.2.0.0/16, version 3
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
```

```
10.3.1.5
65100 65005
    10.3.1.5 from 10.3.1.5 (192.168.2.2)
        Origin IGP, localpref 100, valid, external
65100 65005
    10.3.1.1 from 10.3.1.1 (192.168.2.2)
        Origin IGP, localpref 100, valid, external, best
```

The option is to configure BGP multipath to allow BGP to install both paths in the routing table, as shown in [Example 7-15](#).

Example 7-15. CE1's BGP Multipath Configuration

```
CE1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
CE1(config)#router bgp 65001
CE1(config-router)#maximum-path 2
CE1(config-router)#end
CE1#
CE1#show run | b router bgp
router bgp 65001
  no synchronization
  bgp log-neighbor-changes
  network 10.1.0.0 mask 255.255.0.0
  neighbor 10.3.1.1 remote-as 65100
  neighbor 10.3.1.5 remote-as 65100
  maximum-paths 2
  no auto-summary
!
!Output omitted for brevity
```

Note that after adding the maximum-path configuration, CE1 still marks one path as best, but it also marks both paths as multipath and installs them in the routing table. With eBGP multipath enabled on CE1, CE1 now uses both paths to reach 10.2.4.1.

Example 7-16. Verification of BGP Multipath

Code View: [Scroll](#) / [Show All](#)

```
CE1#show ip bgp 10.2.4.1
BGP routing table entry for 10.2.0.0/16, version 5
Paths: (2 available, best #2, table Default-IP-Routing-Table)
Multipath: eBGP
Flag: 0x800
  Advertised to non-peer-group peers:
    10.3.1.5
    65100 65005
      10.3.1.5 from 10.3.1.5 (192.168.2.2)
          Origin IGP, localpref 100, valid, external, multipath
    65100 65005
      10.3.1.1 from 10.3.1.1 (192.168.2.2)
          Origin IGP, localpref 100, valid, external, multipath, best
CE1#show ip route 10.2.4.1
Routing entry for 10.2.0.0/16
  Known via "bgp 65001", distance 20, metric 0
  Tag 65100, type external
```

```

Last update from 10.3.1.5 00:01:58 ago
Routing Descriptor Blocks:
* 10.3.1.1, from 10.3.1.1, 00:10:41 ago
  Route metric is 0, traffic share count is 1
  AS Hops 2
  10.3.1.5, from 10.3.1.5, 00:01:58 ago
    Route metric is 0, traffic share count is 1
    AS Hops 2
CE1#show ip cef 10.2.4.1
10.2.0.0/16, version 15, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.3.1.1, 0 dependencies, recursive
    traffic share 1
    next hop 10.3.1.1, Serial3/0 via 10.3.1.0/30
    valid adjacency
  via 10.3.1.5, 0 dependencies, recursive
    traffic share 1
    next hop 10.3.1.5, Serial2/0 via 10.3.1.4/30
    valid adjacency
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes

```

This solves the problem of load sharing in the direction of CE to PE. However, an issue still exists in the direction of PE to CE. To allow PE1 to use both paths to reach 10.1.1.1, multipath would also be required on PE1. Remember that with MPLS VPN, all remote locations would point across the core to PE1 to reach CE1 or destinations behind CE1 if this is the only way known. Without multipath enabled, PE1's LFIB shows the information shown in [Example 7-17](#).

Example 7-17. Verification of PE1's LFIB Without Multipath Configuration

```

PE1#show mpls forwarding vrf red 10.1.1.1
Local  Outgoing  Prefix          Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id    switched  interface  interface
19     Untagged   10.1.0.0/16[V]  0         Se2/0      point2point

```

Only one path is known in the LFIB. Look at the routing table and CEF table of PE1 in [Example 7-18](#).

Example 7-18. Verification of PE1's VRF Red Routing and FIB Tables Without Multipath Configuration

```

PE1#show ip cef vrf red 10.1.1.1
10.1.0.0/16, version 11, epoch 0, cached adjacency to Serial2/0
0 packets, 0 bytes
  tag information set
    local tag: 19
  via 10.3.1.2, 0 dependencies, recursive
    next hop 10.3.1.2, Serial2/0 via 10.3.1.0/30
    valid cached adjacency
    tag rewrite with Se2/0, point2point, tags imposed: {}
PE1#show ip route vrf red 10.1.1.1
Routing entry for 10.1.0.0/16
  Known via "bgp 65100", distance 20, metric 0
  Tag 65001, type external
  Last update from 10.3.1.2 00:02:59 ago

```

```
Routing Descriptor Blocks:
* 10.3.1.2, from 10.3.1.2, 00:02:59 ago
  Route metric is 0, traffic share count is 1
  AS Hops 1
```

Only one path is known in the VRF red routing table and the corresponding CEF table. Now, [Example 7-19](#) shows the BGP table for this VRF to verify whether multiple paths are available at that level.

Example 7-19. Verification of BGP Table for VRF Red

```
PE1#show ip bgp vpv4 vrf red 10.1.1.1
BGP routing table entry for 1:1:10.1.0.0/16, version 25296
Paths: (2 available, best #2, table red)
  Advertised to non peer-group peers:
  10.3.1.6 192.168.4.4
  65001
    10.3.1.6 from 10.3.1.6 (10.3.1.6)
      Origin IGP, metric 0, localpref 100, valid, external
      Extended Community: RT:1:1,
      mpls labels in/out 19/nolabel
  65001
    10.3.1.2 from 10.3.1.2 (10.3.1.6)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: RT:1:1,
      mpls labels in/out 19/nolabel
```

Therefore, both paths are in the BGP table, but only one path is available in the routing and CEF tables for VRF red. [Example 7-20](#) shows the configuration for adding multipath for this VRF.

Example 7-20. Multipath Configuration on PE1

```
PE1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
PE1(config)#router bgp 65100
PE1(config-router)#address-family ipv4 vrf red
PE1(config-router-af)#maximum-path 2
```

Now, looking at the BGP table in [Example 7-21](#), both paths are marked multipath.

Example 7-21. Verification of BGP Multipath Configuration on PE1

```
PE1#show ip bgp vpv4 vrf red 10.1.1.1
BGP routing table entry for 1:1:10.1.0.0/16, version 25298
Paths: (2 available, best #2, table red)
Multipath: eBGP
Flag: 0x800
  Advertised to non peer-group peers:
  10.3.1.6 192.168.4.4
  65001
    10.3.1.6 from 10.3.1.6 (10.3.1.6)
      Origin IGP, metric 0, localpref 100, valid, external, multipath
      Extended Community: RT:1:1,
      mpls labels in/out 19/nolabel
  65001
```

```
10.3.1.2 from 10.3.1.2 (10.3.1.6)
  Origin IGP, metric 0, localpref 100, valid, external, multipath, best
  Extended Community: RT:1:1,
  mpls labels in/out 19/nolabel
```

Both paths are in the VRF routing table, VRF CEF table, and LFIB in [Example 7-22](#).

Example 7-22. Verification of Multipath Load Sharing on PE1


Code View: [Scroll](#) / [Show All](#)

```
PE1#show ip route vrf red 10.1.1.1
Routing entry for 10.1.0.0/16
  Known via "bgp 65100", distance 20, metric 0
  Tag 65001, type external
  Last update from 10.3.1.6 00:01:19 ago
  Routing Descriptor Blocks:
  * 10.3.1.2, from 10.3.1.2, 00:05:40 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
  10.3.1.6, from 10.3.1.6, 00:01:19 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
PE1#show ip cef vrf red 10.1.1.1
10.1.0.0/16, version 12, epoch 0, per-destination sharing
0 packets, 0 bytes
  tag information set
  local tag: 19
  via 10.3.1.2, 0 dependencies, recursive
  traffic share 1
  next hop 10.3.1.2, Serial2/0 via 10.3.1.0/30
  valid adjacency
  tag rewrite with Se2/0, point2point, tags imposed: {}
  via 10.3.1.6, 0 dependencies, recursive
  traffic share 1
  next hop 10.3.1.6, Serial3/0 via 10.3.1.4/30
  valid adjacency
  tag rewrite with Se3/0, point2point, tags imposed: {}
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
        internal 0 packets, 0 bytes
PE1#show mpls forwarding vrf red 10.1.1.1
Local   Outgoing   Prefix           Bytes tag   Outgoing   Next Hop
tag     tag or VC  or Tunnel Id     switched   interface
19      Untagged  10.1.0.0/16[V]  0          Se2/0      point2point
        Untagged  10.1.0.0/16[V]  0          Se3/0      point2point
```

With MPLS VPN, you can use the `show ip cef vrf <name> exact-route` command to verify the path that traffic takes on the PE, as shown in [Example 7-23](#).

Example 7-23. Path Verification on PE1 from 10.2.4.1

```
PE1#show ip cef vrf red exact-route 10.2.4.1 10.1.1.1
10.2.4.1      -> 10.1.1.1      : Serial2/0 (next hop 10.3.1.2)
```



A traceroute from the server 10.2.4.1 confirms that traffic from 10.2.4.1 to 10.1.1.1 is taking the path through Serial 2/0 on PE1. Traffic sourced from 10.2.9.3 to 10.1.1.1 takes a different path, as shown in [Example 7-24](#).

Example 7-24. Path Verification on PE1 from 10.2.9.3

```
PE1#show ip cef vrf red exact-route 10.2.9.3 10.1.1.1
10.2.9.3      -> 10.1.1.1      : Serial3/0 (next hop 10.3.1.6)
```

Another way to load-share in this case between the PE and CE routers is to use eBGP multihop. Peering to the loopback addresses through BGP and using statics to reach the loopback address would allow CEF to perform recursion and use both paths.

Using statics or another routing protocol, such as Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), or Routing Information Protocol (RIP), would also be acceptable to enable load sharing in this case. The key is understanding that CEF is a basic building block and that it relies on the routing table. Going from the MPLS network to the CE network (label-to-IP), the PE router has to look into the incoming MPLS packet and perform a hash with the source and destination IP addresses to determine the outgoing interface. Therefore, traffic sharing is still flow based.

Some customers use multilink PPP (MLPPP) instead of CEF to load-share between the PE and CE routers. Check the Feature Navigator or Software Advisor on Cisco.com to verify that support is available for your hardware or software in an MPLS VPN case.

In all cases, adding multiple links between the PE and CE routers can consume memory and CPU resources. Use care when adding links and complexity to avoid network scalability issues.

PE-CE Load Sharing: Site Multihomed to Different PEs

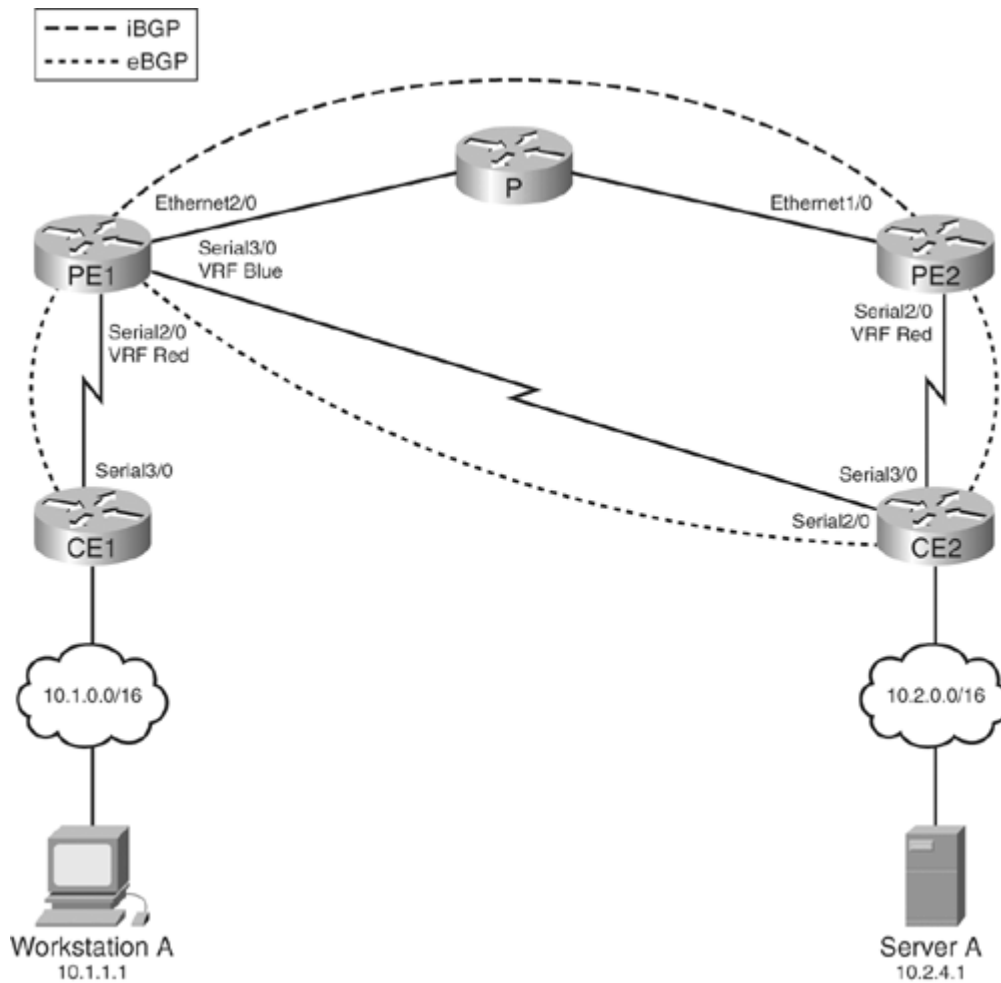
In some cases, it might be necessary to share resources between VPNs with other CE routers connected to multiple PE routers. Importing routes between VRFs enables you to share resources between VPNs by creating an extranet. In such a multihomed environment, it is sometimes convenient to load-share across all known paths. The BGP Multipath Load Sharing feature for both eBGP and iBGP allows configuration for multipath load sharing with both eBGP and iBGP paths in MPLS VPNs. This is called eiBGP multipath.

The eiBGP multipath feature enables core routers to send traffic to destinations through multiple paths using iBGP paths or eBGP paths. In MPLS VPN networks, in a VRF that has paths imported from an eBGP and iBGP path, the PE router can use both eBGP and iBGP paths and install them in the routing table with this feature. Typically, the PEs use only the eBGP paths to reach these CE destinations. Packets sent from the PE router on an eBGP path leave as IP packets, and packets sent on an iBGP path leave as MPLS packets. When the other PE receives MPLS packets, the PE does a label lookup only, so eiBGP multipath does not cause loops. By default, eiBGP multipath performs unequal load sharing by installing multiple paths.

To configure the eiBGP multipath feature, use the `maximum-paths eibgp <n>` command. This command installs up to `n` paths if the first autonomous system (AS) is the same and if the appropriate steps in the path selection algorithm are all equal. The multi-exit discriminator (MED) is not part of the criteria to determine whether a path is suitable for multipath because this is a comparison of different sorts of routes.

In [Figure 7-6](#), for traffic in VRF blue going to 10.2.0.0 through PE1, traffic will always use the eBGP path. In [Example 7-25](#), notice that the eBGP path is the only path known through the LFIB and the VRF routing table.

Figure 7-6. MPLS VPN Extranet
[\[View full size image\]](#)



Example 7-25. Verifying the LFIB and Routing Table for 10.2.0.0

```

PE1#show mpls forwarding vrf blue 10.2.0.0
Local  Outgoing  Prefix          Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id    switched  interface  point2point
20     Untagged   10.2.0.0/16[V]  0         Se3/0      point2point
PE1#show ip route vrf blue 10.2.0.0
Routing entry for 10.2.0.0/16
  Known via "bgp 65100", distance 20, metric 0
  Tag 65005, type external
  Last update from 10.9.1.2 00:04:22 ago
  Routing Descriptor Blocks:
  * 10.9.1.2, from 10.9.1.2, 00:04:22 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1

```

However, looking at the BGP table in [Example 7-26](#), two paths are known: one path through external BGP and the other through the imported path from VRF red learned through internal BGP from PE2.

Example 7-26. Verifying the Number of Paths Known for 10.2.0.0 in VRF Blue

```

PE1#show ip bgp vpn vrf blue 10.2.0.0

```



```
BGP routing table entry for 2:2:10.2.0.0/16, version 9
Paths: (2 available, best #1, table blue)
  Advertised to non peer-group peers:
    192.168.4.4
    65005
      10.9.1.2 from 10.9.1.2 (10.9.1.2)
        Origin IGP, metric 0, localpref 100, valid, external, best
        Extended Community: RT:2:2,
        mpls labels in/out 20/nolabel
    65005, imported path from 1:1:10.2.0.0/16
      192.168.4.4 (metric 21) from 192.168.4.4 (192.168.4.4)
        Origin IGP, metric 0, localpref 100, valid, internal
        Extended Community: RT:1:1,
        mpls labels in/out 20/19
```

By configuring the eIBGP multipath feature, as shown in [Example 7-27](#), Router PE1 can use both paths in VRF blue.

Example 7-27. PE1's eIBGP Configuration

```
PE1(config)#router bgp 65100
PE1(config-router)#address-family ipv4 vrf blue
PE1(config-router-af)#maximum-paths eibgp 2
```

When looking at the BGP output for 10.2.2.0 after configuration, which is shown in [Example 7-28](#), the router marks both paths as multipath, although the eBGP path is best.

Example 7-28. Checking Multipath Configuration

```
PE1#show ip bgp vpn vrf blue 10.2.2.0
BGP routing table entry for 2:2:10.2.0.0/16, version 10
Paths: (2 available, best #1, table blue)
Multipath: eiBGP
  Advertised to non peer-group peers:
    192.168.4.4
    65005
      10.9.1.2 from 10.9.1.2 (10.9.1.2)
        Origin IGP, metric 0, localpref 100, valid, external, multipath, best
        Extended Community: RT:2:2,
        mpls labels in/out 20/nolabel
    65005, imported path from 1:1:10.2.0.0/16
      192.168.4.4 (metric 21) from 192.168.4.4 (192.168.4.4)
        Origin IGP, metric 0, localpref 100, valid, internal, multipath
        Extended Community: RT:1:1,
        mpls labels in/out 20/19
PE1#
```

Inbound IP packets will share the two known paths (iBGP/eBGP) per the show ip cef vrf blue command in [Example 7-29](#).

Example 7-29. Verifying CEF

```
PE1#show ip cef vrf blue 10.2.2.0
10.2.0.0/16, version 11, epoch 0, per-destination sharing
```

```

0 packets, 0 bytes
  tag information set
    local tag: 20
  via 10.9.1.2, 0 dependencies, recursive
    traffic share 1
    next hop 10.9.1.2, Serial3/0 via 10.9.1.0/30
    valid adjacency
    tag rewrite with Se3/0, point2point, tags imposed: {}
  via 192.168.4.4, 0 dependencies, recursive
    traffic share 1
    next hop 172.16.2.2, Ethernet0/0 via 192.168.4.4/32
    valid adjacency
    tag rewrite with Et0/0, 172.16.2.2, tags imposed: {17 19}
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
         internal 0 packets, 0 bytes

```

However, inbound MPLS packets will share the one known eBGP path, as seen in the MPLS forwarding table. Note that only one entry is in the MPLS forwarding table for the VRF in [Example 7-30](#). There will only be one entry if only one eBGP path exists.

Example 7-30. Verifying LFIB

```

PE1#show mpls forwarding vrf blue 10.2.2.0
Local   Outgoing   Prefix           Bytes tag  Outgoing   Next Hop
tag     tag or VC   or Tunnel Id     switched  interface
20      Untagged   10.2.0.0/16[V]   0         Se3/0      point2point

```

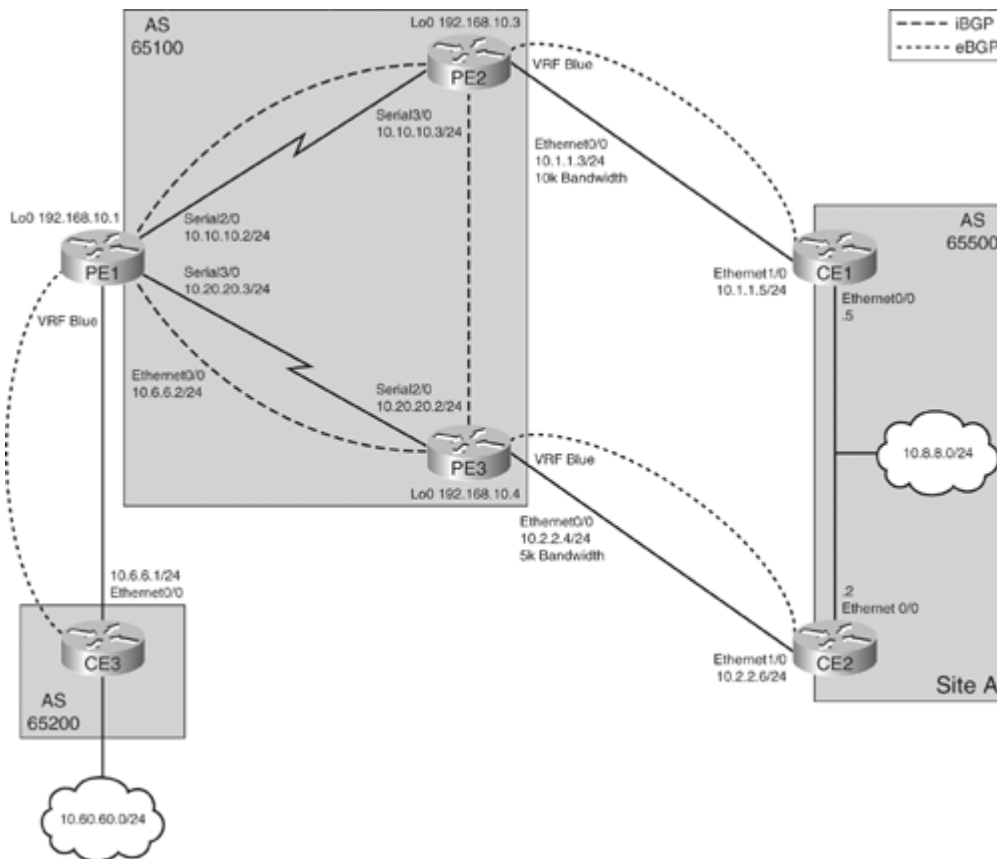
One complication with BGP multipath is that load sharing over links occurs equally, regardless of the different link bandwidths. Unequal load sharing, which occurs with EIGRP, prevents overloading a smaller bandwidth link. Therefore, in Cisco IOS Release 12.2(2)T and later IOS versions, a BGP extended community attribute allows bandwidth differentiation across paths among iBGP peers. Check the Feature Navigator at Cisco.com for availability in your Cisco IOS version and Cisco platform. This feature supports iBGP, eBGP multipath load sharing, and eiBGP multipath load sharing in MPLS VPNs.

For the BGP router to do unequal load sharing between two or more valid BGP paths, the router uses the bandwidth community to set the variance of the links. The `bgp dmzlink-bw` command configures BGP to distribute traffic proportionally to the bandwidth of the link. Each router that contains an external interface that is available for multipath load sharing should have this command enabled. If you do not configure the `bgp dmzlink` command, the routers will send the link bandwidth extended community attribute in the updates to the iBGP peers, but the router will not use the DMZ-link bandwidth locally for unequal-cost load sharing if multiple BGP paths exist.

The `neighbor <address> dmzlink-bw` command propagates the link bandwidth attribute for routes learned from the specified external neighbor in the link bandwidth extended community. When the router receives the link bandwidth attribute and uses the BGP DMZ-link bandwidth, the router sets the traffic load-share value appropriately for multiple paths and passes the value to CEF as with EIGRP unequal-cost load sharing. The multipath router will advertise the aggregated bandwidth to its iBGP peers.

In [Figure 7-7](#), site A has two routers, CE1 and CE2, advertising the prefix 10.8.8.0/24 to their connected provider edge routers, PE2 and PE3, respectively, through external BGP. PE2 and PE3 have a defined bandwidth on the interfaces connected to these CEs. The PE routers carry the BGP DMZ-link bandwidth from the BGP external neighbor through the BGP DMZ-link community. [Example 7-31](#) shows the configuration to propagate the bandwidth attribute on PE2 and PE3.

Figure 7-7. BGP DMZ Link
[View full size image]



Example 7-31. Modified Configuration on PE2 and PE3 to Carry BGP DMZ-Link Community

Code View: [Scroll](#) / [Show All](#)

```

303-PE2#show run | b router bgp
router bgp 65100
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.10.1 remote-as 65100
 neighbor 192.168.10.1 update-source Loopback0
 no auto-summary
 !
 address-family vpv4
 neighbor 192.168.10.1 activate
 neighbor 192.168.10.1 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf blue
 redistribute connected
 neighbor 10.1.1.5 remote-as 65500
 neighbor 10.1.1.5 activate
 neighbor 10.1.1.5 dmzlink-bw
 no auto-summary
 no synchronization
 exit-address-family
    
```

```

!
PE3-304#show run | b router bgp
router bgp 65100
  no synchronization
  bgp log-neighbor-changes
  neighbor 192.168.10.1 remote-as 65100
  neighbor 192.168.10.1 update-source Loopback0
  no auto-summary
!
  address-family vpnv4
  neighbor 192.168.10.1 activate
  neighbor 192.168.10.1 send-community extended
  exit-address-family
!
  address-family ipv4 vrf blue
  redistribute connected
  neighbor 10.2.2.6 remote-as 65500
  neighbor 10.2.2.6 activate
  neighbor 10.2.2.6 dmzlink-bw
  no auto-summary
  no synchronization
  exit-address-family

```

The output in [Example 7-32](#) shows that the PE routers carry the bandwidth. The router expresses the DMZ-link bandwidth in kilobytes (KBps), whereas the interface bandwidth is in kilobits (kbps).

Example 7-32. Verifying Bandwidth Carried Through BGP

```

303-PE2#show ip bgp vpn vrf blue 10.8.8.0
BGP routing table entry for 100:100:10.8.8.0/24, version 13
Paths: (1 available, best #1, table blue)
  Advertised to update-groups:
    1
    65500
    10.1.1.5 (via blue) from 10.1.1.5 (10.8.8.5)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: RT:100:100
      DMZ-Link Bw 1250 kbytes

PE3-304#show ip bgp vpn vrf blue 10.8.8.0
BGP routing table entry for 100:100:10.8.8.0/24, version 6
Paths: (1 available, best #1, table blue)
  Advertised to update-groups:
    1
    65500
    10.2.2.6 (via blue) from 10.2.2.6 (10.8.8.6)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: RT:100:100
      DMZ-Link Bw 625 kbytes
PE3-304#

```

Before the availability of the iBGP load-sharing feature, Cisco IOS would only select one iBGP best path. For iBGP load sharing to be successful, by default, the routes must have equal attributes, such as weight, local

preference, AS path, origin code, and IGP metric. If the attributes are not the same, the router does not use the path as a multiple path and only picks the best path.

In this case, PE1's configuration allows utilization of both paths through iBGP load sharing, and the attributes are the equal. Therefore, PE1 receives the route and uses both paths through multipath configuration. PE1 uses the defined DMZ-link bandwidth locally because of the configured `bgp dmzlink-bw` command under the address family VPNv4.

Example 7-33. Configuring PE1 to Receive and Use the BGP-Carried Bandwidth

Code View: [Scroll](#) / [Show All](#)

```
302-PE1#show run | b router bgp
router bgp 65100
  no synchronization
  bgp log-neighbor-changes
  neighbor 192.168.10.3 remote-as 65100
  neighbor 192.168.10.3 update-source Loopback0
  neighbor 192.168.10.4 remote-as 65100
  neighbor 192.168.10.4 update-source Loopback0
  no auto-summary
  !
  address-family vpnv4
  neighbor 192.168.10.3 activate
  neighbor 192.168.10.3 send-community extended
  neighbor 192.168.10.4 activate
  neighbor 192.168.10.4 send-community extended
  bgp dmzlink-bw
  exit-address-family
  !
  address-family ipv4 vrf blue
  neighbor 10.6.6.1 remote-as 65200
  neighbor 10.6.6.1 activate
  maximum-paths ibgp 2
  no auto-summary
  no synchronization
  exit-address-family
```

Router PE1 receives two paths, including the DMZ-link bandwidth, through the output shown in [Example 7-34](#).

Example 7-34. BGP Verification That PE1 Receives Paths and Bandwidth

```
302-PE1#show ip bgp vpn vrf blue 10.8.8.0
BGP routing table entry for 100:100:10.8.8.0/24, version 11
Paths: (2 available, best #1, table blue)
Multipath: iBGP
  Advertised to update-groups:
    1
  65500
    192.168.10.3 (metric 65) from 192.168.10.3 (192.168.10.3)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath, best
      Extended Community: RT:100:100
      DMZ-Link Bw 1250 kbytes
  65500
    192.168.10.4 (metric 65) from 192.168.10.4 (192.168.10.4)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community: RT:100:100
```

```
DMZ-Link Bw 625 kbytes
302-PE1#
```

PE1 passes this DMZ-link bandwidth to the routing table and VRF table, as shown in [Example 7-35](#). Notice that the traffic count is two for the path through 192.168.10.3 and one for the path through 192.168.10.4.

Example 7-35. Verifying BGP Unequal Load Sharing Through Routing and CEF Tables

Code View: [Scroll](#) / [Show All](#)

```
302-PE1#show ip route vrf blue 10.8.8.0
Routing entry for 10.8.8.0/24
  Known via "bgp 65100", distance 200, metric 0
  Tag 65500, type internal
  Last update from 192.168.10.4 04:09:31 ago
  Routing Descriptor Blocks:
  * 192.168.10.3 (Default-IP-Routing-Table), from 192.168.10.3, 04:09:31
ago
    Route metric is 0, traffic share count is 2
    AS Hops 1, BGP network version 0
    Route tag 65500
  192.168.10.4 (Default-IP-Routing-Table), from 192.168.10.4, 04:09:31
ago
    Route metric is 0, traffic share count is 1
    AS Hops 1, BGP network version 0
    Route tag 65500
302-PE1#show ip cef vrf blue 10.8.8.0
10.8.8.0/24, version 82, epoch 0, per-destination sharing
0 packets, 0 bytes
  tag information set, all rewrites owned
  local tag: VPN route head
  via 192.168.10.3, 0 dependencies, recursive
  traffic share 2
  next hop 10.10.10.3, Serial2/0 via 192.168.10.3/32 (Default)
  valid adjacency
  tag rewrite with
    Recursive rewrite via 192.168.10.3/32, tags imposed {25}
  via 192.168.10.4, 0 dependencies, recursive
  traffic share 1
  next hop 10.20.20.4, Serial3/0 via 192.168.10.4/32 (Default)
  valid adjacency
  tag rewrite with
    Recursive rewrite via 192.168.10.4/32, tags imposed {18}
0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
        internal 0 packets, 0 bytes
302-PE1#
```

Load Sharing Between P and P Devices

Sometimes network architects desire multiple links between the P and P devices to help distribute load. Different configurations are possible. Multiple links could exist between the same P devices, or multiple links could exist to different P devices. When going from one P device to another, the P device bases the forwarding decision on the MPLS label. With multiple paths, CEF is still the basis for populating the load-sharing table and deciding the

ultimate path to a destination. When a P router with multiple outgoing equal-cost paths receives a packet with an MPLS label, it makes the load-sharing decision using the hash of the source and destination IP addresses as inputs in the CEF algorithm. The output of the hash determines which interface and label to use.

On a P router, using the `show ip cef exact-route` command is not applicable because it requires the presence of the destination IP addresses in the CEF table. As a quick, rough check of load sharing on a P router, an administrator can check the LFIB counters. However, this might not be accurate depending on the design. In later Cisco IOS code, such as IOS Release 12.0(28)S2, the `show mpls forwarding-table labels <label> exact-path ipv4 <source ip> <destination ip>` command is available to check the path.

Table 7-11 describes the commands used to determine the CEF path.

Table 7-11. Commands to Determine the CEF Path for a Flow		
Device Function	Most IOS Routers	SUP720
IP-to-IP routing	<code>show ip cef exact-route</code>	<code>show mpls cef exact-route</code>
IP-to-label switching	<code>show ip cef vrf <vrf> exact-route</code>	<code>show mpls cef exact-route <vrf></code>
Label-to-label switching	<code>show mpls forwarding-table labels <label> exact-path ipv4 <src ip> <dst ip></code>	—

CEF and MPLS VPN Load-Sharing Platform Dependencies

You must consider some platform dependencies when doing load sharing in an MPLS VPN scenario. On Cisco 12000 series routers, each line card makes independent forwarding decisions using hardware-based CEF tables. Therefore, the route calculated in software and viewed using the `show ip cef exact-route <src> <dst>` command executed on the RP might not be the actual path taken.

Another point is that the Engine 4 MPLS load-sharing mechanism is different from other line cards. The hash uses the destination next-hop address to load-share on the E4 line card instead of the source-destination pair. Therefore, load sharing can occur across the links in the core based on egress PE loopback addresses in an MPLS VPN environment. On E4+, load sharing occurs based on the IP source and destination addresses on a per-flow basis and can be more granular.

Engine 3 only supports per-destination load sharing as an ingress card. For IP packets, the line card hardware can do a hash on the source and destination IP addresses to determine the egress interface. If an Engine 3 port is a VRF interface, the line card's software instead of the hardware does the load-sharing hash. Therefore, the software selects the egress interface on a round-robin basis and programs the hardware in this case.

Originally, on a Cisco Catalyst 6500 with a SUP720, load sharing was available only in the IP-to-label and the label-to-IP directions for MPLS VPN. On a Catalyst 6500 with a SUP720 or a Cisco 7600 router that serves as a P router, the `mpls load-balance per-label` command allows use of the incoming label for load sharing. With this command, load sharing occurs based on the topmost incoming label in the stack that the packet arrives with. So, for example, if the outgoing path is an equal-cost path, load sharing will occur using the top IGP label with this feature enabled.