CCIE Professional Development

Inside Cisco IOS Software Architecture

An essential guide to understanding the internal operation of Cisco routers

Vijay Bollapragada, CCIE® No. 1606
Curtis Murphy, CCIE No. 1521
Russ White, CCIE No. 2635

ciscopress.com

Inside Cisco IOS Software Architecture

Vijay Bollapragada, CCIE #1606,
Curtis Murphy, CCIE #1521, & Russ White, CCIE #2635

Cisco Press

Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

Optimum switching is still similar to fast switching in many ways, including:

- Cache entries are built when the first packet is process switched toward a destination.
- Cache entries are aged and invalidated as the routing table or other cache information changes.
- Load sharing still occurs based strictly on the destination address.
- The same rules are still used to decide what cache entry to build for each particular destination.

The output of **show ip cache optimum**, shown in Example 2-5, is very similar to the output of **show ip cache verbose** in Example 2-3. The primary differences are in the first few lines of output, which provide information about the mtree in which the information is stored.

Example 2-5 *Displaying Optimum Cache Entries*

```
router#show ip cache optimum
Optimum Route Cache
 1 prefixes, 1 nodes, 0 leaf refcount, 8K bytes
 0 nodes pending, 0 node alloc failures
 8 prefix updates, 4 prefix invalidations

Prefix/Length    Age    Interface    Next Hop
10.1.1.16/32-24   1w4d   Ethernet0    10.1.1.16
```

The first few lines of information, under the heading **Optimum Route Cache**, note the number of nodes in the mtree, the number of leaf nodes that are referred to by other nodes (because of recursive routing), the memory allocation information, and the number of times the mtree has been updated.

The Optimum Cache mtree doesn't start with all its nodes populated; most of the children under each node will be NULL (or lead to nowhere). As the cache is built through the process switching of packets, it fills up, although you would probably never encounter an Optimum Cache mtree with all the nodes filled.

Cisco Express Forwarding

Cisco Express Forwarding (CEF) is the newest, and fastest, switching method available in IOS. CEF was invented to help overcome the major deficiencies of the fast switching method. Reviewing the description of fast switching for a moment, the major deficiencies noted were:

- Lack of support for overlapping cache entries.
- Any change in the routing table or ARP cache results in the invalidation of large sections of the route cache, because there is little or no correlation between the route cache and the tables on which the cache is based.

- The first packet to any given destination must be process switched to build a route cache entry.
- Inefficient load balancing in some situations (primarily in cases where many hosts are transacting with one server).

Most of these deficiencies do not pose a problem in the average enterprise network because routes don't change often and routing tables aren't extremely large. However, there is one environment where these shortcomings *do* become a problem: the Internet backbone.

Internet backbone routers have extremely large routing tables, about 56,000 routes in early 1999 and still growing. Some Internet backbone routers actually carry more than 100,000 routes. The routing table also changes constantly, causing cache entries to be invalidated frequently. In fact, entries are invalidated often enough to cause a significant portion of traffic through some Internet routers to be process switched. CEF was developed specifically to improve routing performance in these types of environments.

CEF was initially tested in the large-scale network environment of the Internet. Internet service providers were given a series of special IOS software images (nicknamed the *ISP Geeks images*) with CEF to see how it operated under extreme conditions. The technology proved itself capable of handling the Internet backbone workload and was eventually integrated into the Cisco IOS product, becoming the default switching mode in Cisco IOS Release 12.0. It is now the only switching mode available on some platforms—in particular, the Cisco 12000 and the Catalyst 8500.

How CEF Works

Unlike fast switching, which builds a cached subset of the routing table and MAC address tables, CEF builds its own structures that mirror the entire routing and MAC address tables. There are two major structures maintained by CEF. These structures collectively can be considered the CEF "Fast Cache":

- CEF table
- Adjacency table

The CEF Table

The CEF table is a "stripped-down" version of the routing table, implemented as a 256-way mtrie data structure for optimum retrieval performance. You can display the size of the CEF table (and other general information about the table) by using the command **show ip cef summary**, as demonstrated in Example 2-6.

Example 2-6 *Displaying CEF Table Information*

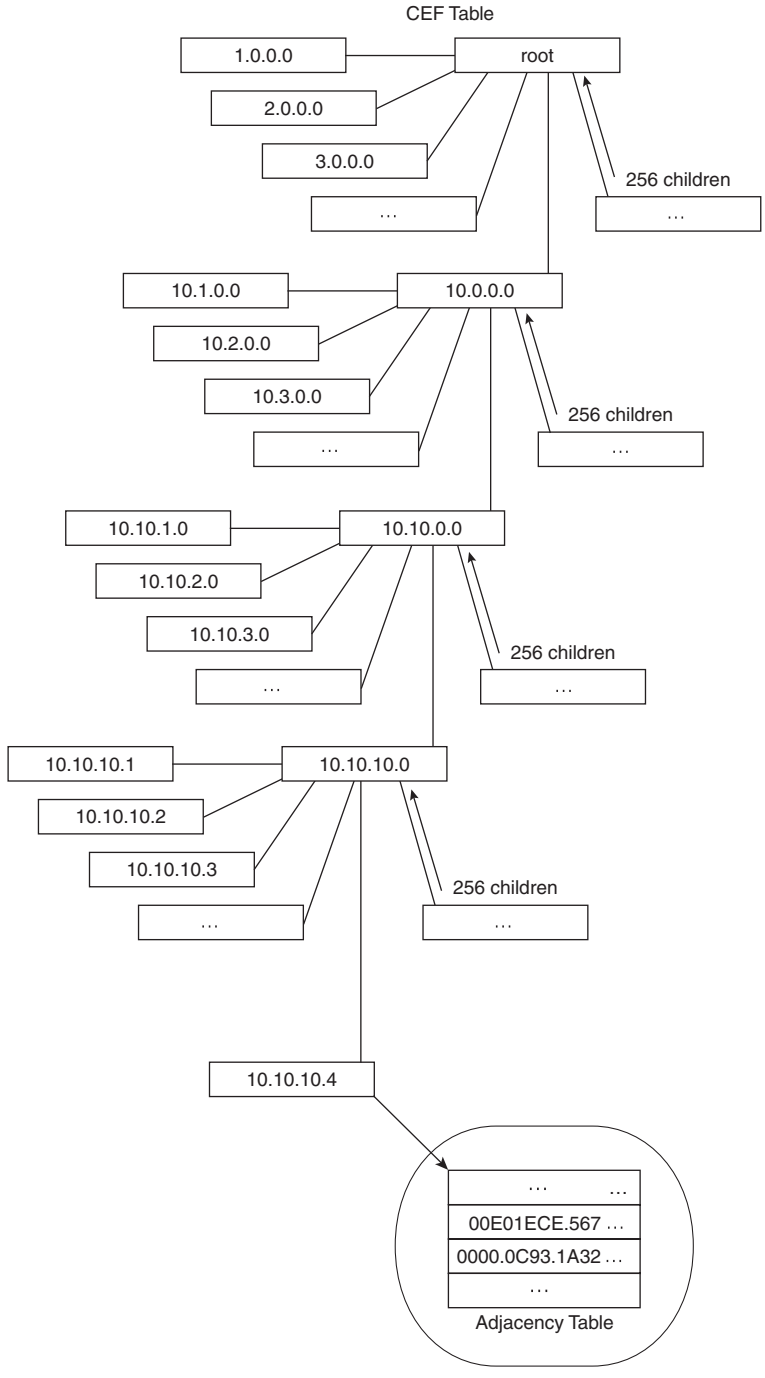
```
router#show ip cef summary
IP Distributed CEF with switching (Table Version 96)
  33 routes, 0 reresolve, 0 unresolved (0 old, 0 new)
  33 leaves, 31 nodes, 36256 bytes, 96 inserts, 63 invalidations
  1 load sharing elements, 328 bytes, 2 references
  1 CEF resets, 8 revisions of existing leaves
  refcounts:  8226 leaf, 8192 node

The adjacency table has five adjacencies.
```

In a 256-way mtrie structure, each node in the structure can have up to 256 children. In the CEF table, each child (or link) is used to represent a different address in one octet of an IP address, as shown in Figure 2-10.

For example, given the IP address 10.10.10.4, you would locate the data by choosing the tenth child off the root, the tenth child off that node, the tenth child off that node, and then, finally, the fourth child off the last node. This final node, or *end node*, contains a pointer to an entry in another table outside the CEF, called the *adjacency table*. The adjacency table actually contains the MAC header and other information needed to switch the packet.

NOTE An *mtree* data structure stores the actual data within the tree structure itself; for example, in the optimum switching mtree cache, the MAC header data used for forwarding packets is actually stored inside the mtree. In an *mtrie* data structure, the tree structure is used to locate the desired data, but the data itself is stored elsewhere.

Figure 2-10 *CEF mtrie and Adjacency Table*



Adjacency Table

The adjacency table contains the MAC-layer packet-header data for directly connected next hops. The adjacency table is populated with data from the ARP table, the Frame Relay map table, and other tables of this type. The **show adjacency** command displays the adjacency table, as demonstrated in Example 2-7.

Example 2-7 *Displaying Adjacency Table Information*

```
router#show adjacency
Protocol Interface                 Address
IP       POS0/0/0                  point2point(15)
IP       Serial1/0/0               point2point(5)
IP       FastEthernet6/0/0         17.1.1.2(16)
IP       Ethernet4/0               10.105.1.1(9)
IP       Ethernet4/0               10.105.1.179(5)
Router#
```

There are several types of adjacency entries you might see in the **Address** column of **show adjacency**:

- **Cached adjacency**—A prebuilt MAC header for the next hop toward this destination.
- **Punt**—Packets destined to this address must be passed up to the next switching path to be switched.
- **Host-route**—This destination is a directly connected host.
- **Drop**—Packets destined to this address are dropped.
- **Incomplete**—The MAC-layer header for this destination is incomplete; typically, this means the ARP cache entry for this destination is incomplete or otherwise malformed.
- **Glean**—This is a directly attached destination, but there is no pre-built MAC-layer header; an ARP request needs to be sent so a MAC-layer header can be built.

The CEF Method and Its Advantages

Like fast switching, CEF switching utilizes cache entries to perform its switching operation entirely during a processor interrupt. A major difference between fast switching and CEF switching, however, is the point at which the cache entries are built. While fast switching requires the first packet to any given destination be process switched to build a cache entry, CEF does not. Instead, the CEF table is built directly from the routing table and the adjacency table is built directly from the ARP cache. These CEF structures are built before any packets are switched.

Because CEF pre-builds the cache before any packets are switched, every packet received for a reachable destination can be forwarded by the IOS during the receive interrupt. It is never necessary to process switch a packet to get a cache entry built. Pre-building the cache greatly improves the routing performance on routers with large routing tables. With normal fast switching, IOS can be overwhelmed with process-level traffic before the route cache entries are created. With CEF, this massive process-level load is eliminated, preventing the IOS from choking on the process-level switching bottleneck when network routes are flapping.

Splitting the reachability/interface data and the MAC-layer header data into two structures also provides other advantages: Because the tables used in switching packets are directly related to their information sources, the cache doesn't need an aging process anymore.

Entries in the CEF structures never age out. Any changes in the routing table or the ARP cache can be easily reflected in the CEF structures. Also, with CEF there is no longer any need to invalidate a large number of entries when the routing table or the ARP cache change.

Traffic Load Sharing with CEF

Traffic load sharing with CEF switching can be either per source/destination pair (the default) or per packet. Traffic sharing based on the source/destination pair resolves the problems described in the earlier fast switching example, where all the traffic destined to the server would take one link because the cache is built on a per-destination basis. How does it do this? Instead of pointing to an entry in the adjacency table, an entry in the CEF table can also point to a *load share* structure, as shown in Figure 2-11.

When the CEF switching method finds a load-share entry—rather than an adjacency table entry—as the result of a CEF table search, it uses the source and destination address to decide which of the entries to use from the load-share table. Each entry in the load-share table then points to an adjacency table entry that contains the MAC header and other information needed to forward the packet.