How to Choose the Best Router Switching Path for Your Network

Help us help you.

Please rate this document.

- Excellent
- Good
- Average
- Fair
- Poor

This document solved my problem.

- Yes
- No
- Just browsing

Suggestions for improvement:

(256 character limit)

Optional contact information:

Name:

Email:

Send

Contents

Introduction

There are a plethora of switching paths available to various Cisco routers and Cisco IOS releases. Which is the best one for your network, and how do they all work? This white paper is an attempt to explain each of the following switching paths so you can make the best decision about which switching path fits your network.

To begin, let's examine the forwarding process itself. There are three steps to forwarding a packet through a router:

1. Determine if the packet's destination is reachable.
2. Determine the next hop toward the destination, and the interface through which that next hop is reachable.
3. Rewrite the Media Access Control (MAC) header on the packet so it will successfully reach its next hop.

Each of these steps is critical for the packet to reach its destination.

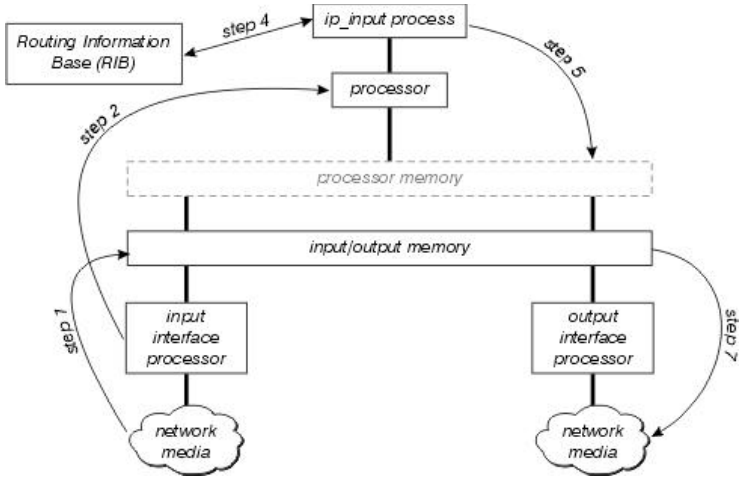
Note: Throughout this document, we use the IP switching path as an example; virtually all the information provided here is applicable to equivalent switching paths for other protocols, if they exist.

Process Switching

Process switching is the lowest common denominator in switching paths; it's available on every version of IOS, on every platform, and for every type of traffic being switched. Process switching is defined by two essential concepts:

- The forwarding decision and information used to rewrite the MAC header on the packet are taken from the routing table (from the routing informati on base, or RIB) and the Address Resolution Protocol (ARP) cache, or from some other table that contains the MAC header information mapped to the IP address of each host that is directly connected to the router.
- The packet is switched by a normal process running within IOS. In other words, the forwarding decision is made by a process scheduled through the IOS scheduler and running as a peer to other processes on the router, such as routing protocols. Processes that normally run on the router aren't interrupted to process switch a packet.

The figure below illustrates the process switching path.



Let's examine this diagram in more detail:

1. The interface processor first detects there is a packet on the network media, and transfers this packet to the input/output memory on the router.

2. The interface processor generates a receive interrupt. During this interrupt, the central processor determines what type of packet this is (we'll assume it's an IP packet), and copies it into processor memory if necessary (this decision is platform dependent). Finally, the processor places the packet on the appropriate process' input queue and the interrupt is released.

3. The next time the scheduler runs, it notes the packet in the input queue of **ip_input**, and schedules this process to run.

4. When **ip_input** runs, it consults the RIB to determine the next hop and the output interface, then consults the ARP cache to determine the correct physical layer address for this next hop.

5. **ip_input** then rewrites the packet's MAC header, and places the packet on the output queue of the correct outbound interface.

6. The packet is copied from the output queue of the outbound interface to the transmit queue of the outbound interface; any outbound quality of service takes place between these two queues.

7. The output interface processor detects the packet on its transmit queue, and transfers the packet onto the network media.

Almost all features that effect packet switching, such as Network Address Translation (NAT) and Policy Routing, make their debut in the process switching path. Once they have been proven, and optimized, these features may, or may not, appear in interrupt context switching.

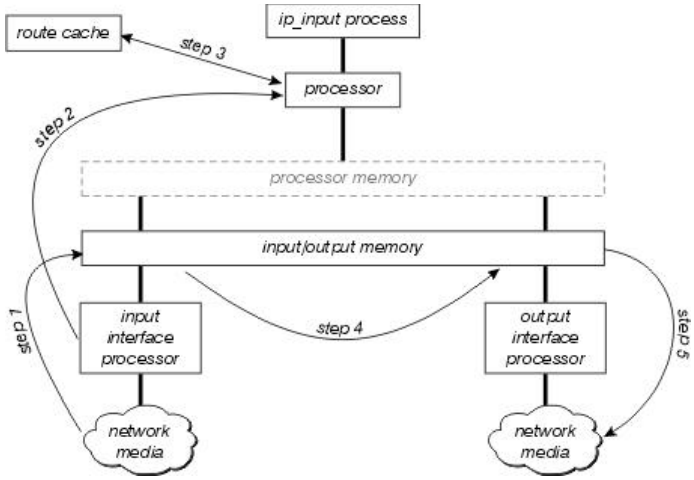
Interrupt Context Switching

Interrupt context switching is the second of the primary switching methods used by Cisco routers. The primary differences between interrupt context switching and process switching are:

- The process currently running on the processor is interrupted to switch the packet. Packets are switched on demand, rather than switched only when the **ip_input** process can be scheduled.

- The processor uses some form of route cache to find all the information needed to switch the packet.

The following figure illustrates interrupt context switching.



Let's examine this diagram in more detail:

1. The interface processor first detects there is a packet on the network media, and transfers this packet to the input/output memory on the router.

2. The interface processor generates a receive interrupt. During this interrupt, the central processor determines what type of packet this is (we'll assume it's an IP packet), and then begins to switch the packet.

3. The processor searches the route cache to determine if the packet's destination is reachable, what the output interface should be, what the next hop towards this destination is, and finally, what MAC header the packet should have to successfully reach the next hop. The processor uses this information to rewrite the packet's MAC header.

4. The packet is now copied to either the transmit or output queue of the outbound interface (depending on various factors). The receive interrupt now returns, and the process that was running on the processor before the interrupt occurred continues running.

5. The output interface processor detects the packet on its transmit queue, and transfers the packet onto the network media.

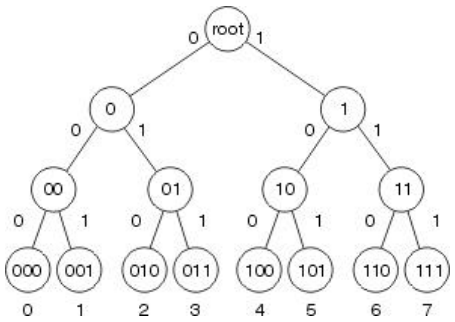
The first question that comes to mind after reading this description is "What's in the cache?" There are three possible answers, depending on the type of interrupt context switching:

- Fast Switching
- Optimum Switching
- Cisco Express Forwarding

We'll look at each of these route cache types (or switching paths) one at a time.

Fast Switching

Fast switching stores the forwarding information and MAC header rewrite string using a binary tree for quick lookup and reference. The following figure illustrates a binary tree.



In Fast Switching, the reachability information is indicated by the existence of a node on the binary tree for the destination of the packet. The MAC header and outbound interface for each destination are stored as part of the node's information within the tree. The binary tree can actually have 32 levels—the tree above is extremely abbreviated for the purpose of illustration.

To search a binary tree, you simply start from the left (with the most significant digit) in the (binary) number you are looking for, and branch right or left in the tree based on that number. For instance, if you're looking for the information related to the number 4 in this tree, you would begin by branching right, because the first binary digit is 1. You would follow the tree down, comparing the next digit in the (binary) number, until you reach the end.

Characteristics of the Fast Switching

Fast Switching has several characteristics that are a result of the binary tree structure and the storage of the MAC header rewrite information as part of the tree nodes.

- Since there is no correlation between the routing table and the fast cache contents (MAC header rewrite, for example), building cache entries involves all the processing that must be done in the process switching path. Therefore, fast cache entries are built as packets are process switched.
- Since there is no correlation between the MAC headers (used for rewrites) in the ARP cache and the structure of the fast cache, when the ARP table changes, some portion of the fast cache must be invalidated (and recreated through the process switching of packets).
- The fast cache can only build entries at one depth (one prefix length) for any particular destination within the routing table.
- There is no way to point from one entry to another within the fast cache (the MAC header and outbound interface information are expected to be within the node), so all routing recursions must be resolved while a fast cache entry is being built. In other words, recursive routes can't be resolved within the fast cache itself.

Aging Fast Switching Entries

To keep the fast switching entries from losing their synchronization with the routing table and ARP cache, and to keep unused entries in the fast cache from unduly consuming memory on the router, 1/20th of the fast cache is invalidated, randomly, every minute. If the routers memory drops below a very low watermark, 1/5th of the fast cache entries are invalidated every minute.

Fast Switching Prefix Length

What prefix length does the fast switching build entries for if it can only build to one prefix length for every destination? Within the terms of the fast switching, a destination is a single reachable destination within the routing table, or a major network. The rules for deciding what prefix length to build a given cache entry are:

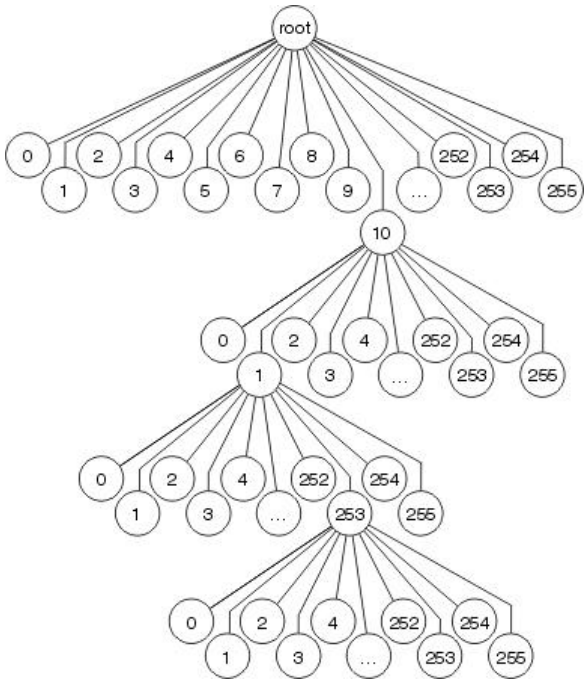
- If building a fast policy entry, always cache to /32.
- If building an entry against an Multiprotocol over ATM virtual circuit (MPOA VC), always cache to /32.
- If the network is not subnetted (it's a major network entry):
  - If it's directly connected, use /32;
  - Otherwise use the major net mask.
- If it's a supernet use the supernet's mask.
- If the network is subnetted:
  - If directly connected, use /32;
  - If there are multiple paths to this subnet, use /32;
  - In all other cases, use longest prefix length in this major net.

Load Sharing

Fast switching is entirely destination based; load sharing occurs on a per-destination basis. If there are multiple equal cost paths for a particular destination network, fast cache has one entry for each host reachable within that network, but all traffic destined to a particular host follows one link.

Optimum Switching

Optimum switching stores the forwarding information and the MAC header rewrite information in a 256 way multiway tree (256 way mtree). Using an mtree reduces the number of steps which must be taken when looking up a prefix, as illustrated in the next figure.

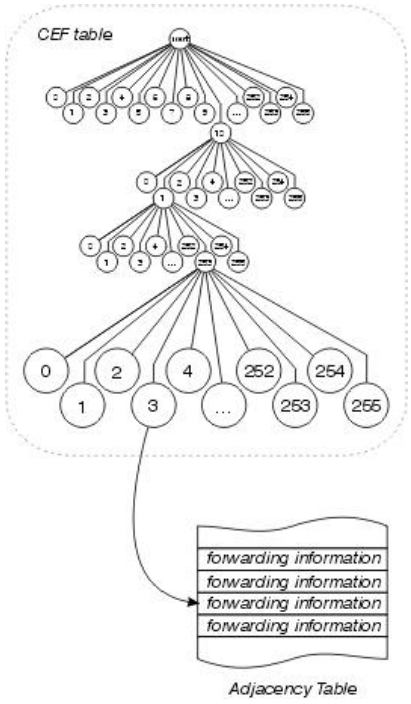


Each octet is used to determine which of the 256 branches to take at each level of the tree, which means there are, at most, 4 lookups involved in finding any destination. For shorter prefix lengths, only one-three lookups may be required. The MAC header rewrite and output interface information are stored as part of the tree node, so cache invalidation and aging still occur as in the fast switching.

Optimum Switching also determines the prefix length for each cache entry in the same way as fast switching.

Cisco Express Forwarding

Cisco Express Forwarding (CEF), also uses a 256 way data structure to store forwarding and MAC header rewrite information, but it doesn't use a tree. CEF uses a trie, which means the actual information being searched for isn't in the data structure; instead, the data is stored in a separate data structure, and the trie simply points to it. In other words, rather than storing the outbound interface and MAC header rewrite within the tree itself, CEF stores this information in a separate data structure called the adjacency table.



This separation of the reachability information (in the CEF table) and the forwarding information (in the adjacency table), provides a number of benefits:

- The adjacency table can be built separately from the CEF table, allowing both to build without process switching any packets.
- The MAC header rewrite used to forward a packet isn't stored in cache entries, so changes in a MAC header rewrite string don't require invalidation of cache entries.
- Recursive routes can be resolved by pointing to the recursed next hop, rather than directly to the forwarding information.

Essentially, all cache aging is eliminated, and the cache is pre-built based on the information contained in the routing table and ARP cache. There is no need to process switch any packet to build a cache entry.

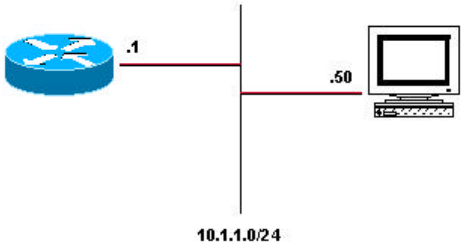
Other Entries in the Adjacency Table

The adjacency table can contain entries other than MAC header rewrite strings and outbound interface information. Some of the various types of entries that can be placed in the adjacency table include:

- **cache** A MAC header rewrite string and outbound interface used to reach a particular adjacent host or router.
- **receive** Packets destined to this IP address should be received by the router. This includes broadcast addresses and addresses configured on the router itself.
- **drop** Packets destined to this IP address should be dropped. This could be used for traffic denied by an access list, or routed to a NULL interface.
- **punt** CEF cannot switch this packet; pass it to the next best switching method (generally fast switching) for processing.
- **glean** The next hop is directly attached, but there are no MAC header rewrite strings currently available.

Glean Adjacencies

A glean adjacency entry indicates that a particular next hop should directly connected, but there is no MAC header rewrite information available. How do these get built and used? A router running CEF and attached to a broadcast network, as shown in the figure below, builds a number of adjacency table entries by default.



The four adjacency table entries built by default are:

```
10.1.1.0/24, version 17, attached, connected
0 packets, 0 bytes
  via Ethernet2/0, 0 dependencies
    valid glean adjacency
10.1.1.0/32, version 4, receive
10.1.1.1/32, version 3, receive
10.1.1.255/32, version 5, receive
```

Note there are four entries: three receives, and one glean. Each receive entry represents a broadcast address or an address configured on the router, while the glean entry represents the remainder of the address space on the attached network. If a packet is received for host 10.1.1.50, the router attempts to switch it, and finds it resolved to this glean adjacency. CEF then signals that an ARP cache entry is needed for 10.1.1.50, the ARP process sends an ARP packet, and the appropriate adjacency table entry is built from the new ARP cache information. After this step is complete, the adjacency table has an entry for 10.1.1.50.

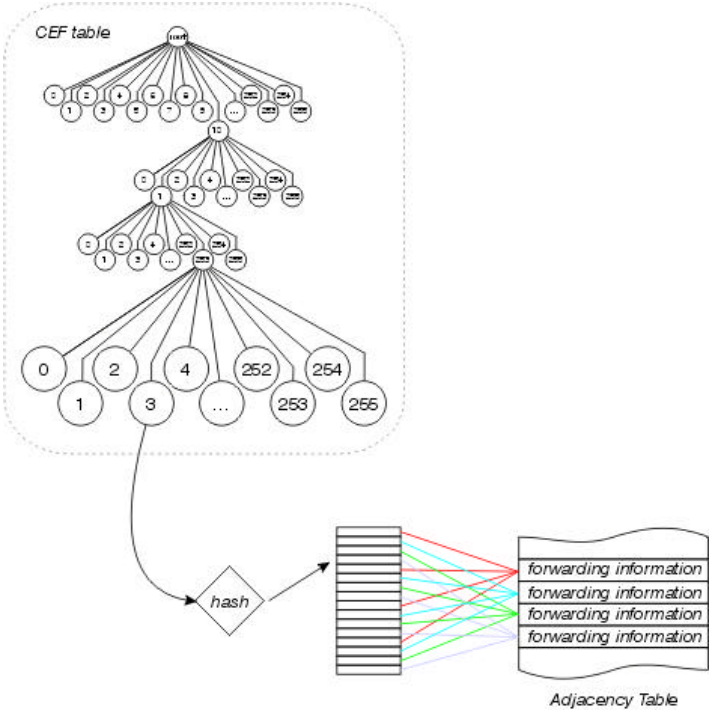
```
10.1.1.0/24, version 17, attached, connected
0 packets, 0 bytes
```

```
    via Ethernet2/0, 0 dependencies
      valid glean adjacency
  10.1.1.0/32, version 4, receive
  10.1.1.1/32, version 3, receive
  10.1.1.50/32, version 12, cached adjacency 208.0.3.2
  0 packets, 0 bytes
    via 208.0.3.2, Ethernet2/0, 1 dependency
      next hop 208.0.3.2, Ethernet2/0
      valid cached adjacency
  10.1.1.255/32, version 5, receive
```

The next packet the router receives destined for 10.1.1.50 is switched through this new adjacency.

Load Sharing

CEF also takes advantage of the separation between the CEF table and the adjacency table to provide a better form of load sharing than any other interrupt context switching mode. A loadshare table is inserted between the CEF table and the adjacency table, as illustrated in the figure below.



The CEF table points to this loadshare table, which contains pointers to the various adjacency table entries for available parallel paths. The source and destination addresses are passed through a hash algorithm to determine which loadshare table entry to use for each packet. Per packet load sharing can be configured, in which case each packet uses a different loadshare table entry.

Each loadshare table has 16 entries among which the paths available are divided based on the traffic share counter in the routing table. If the traffic share counters in the routing table are all 1 (as in the case of multiple equal cost paths), each possible next hop receives an equal number of pointers from the loadshare table. If the number of available paths isn't evenly divisible into 16 (since there are 16 loadshare table entries), some paths will have more entries than others.

Beginning in IOS 12.0, the number of entries in the loadshare table is reduced to make certain each path has a proportionate number of loadshare table entries. For instance, if there are three equal cost paths in the routing table, only 15 loadshare table entries are used.

Which Switching Path Is Best?

Whenever possible, you want your routers to be switching in the interrupt context because it's at least an order of a magnitude faster than process level switching. CEF switching is definitely faster and better than any other switching mode. We recommend you use CEF if the protocol and IOS you're running supports it. This is particularly true if you have a number of parallel links across which traffic should be load shared.

Related Information

- **IP Routing Technical Tips**
- **Routing Protocol Technical Tips**
- **IP Routing Protocols Top Issues**
- **IP Routing Protocols Support Pages**

Cisco Systems TAC Certified
Learn more about Cisco TAC Certification...

All contents are Copyright © 1992--2001 Cisco Systems Inc. All rights reserved. Important Notices and Privacy Statement.