Implementing Cisco Express Forwarding

Implementing Cisco Express Forwarding

Cisco Express Forwarding (CEF) is an advanced, Layer 3 IP switching technology. CEF optimizes network performance and scalability for networks with large and dynamic traffic patterns, such as the Internet, on networks characterized by intensive web-based applications, or interactive sessions. CEF is an inherent feature and the users need not perform any configuration to enable it. If required, the users can change the default route purge delay and static routes.

Components

Cisco IOS XR software CEF always operates in CEF mode with two distinct components:

- Forwarding Information Base (FIB) database: The protocol-dependent FIB process maintains the forwarding tables for IPv4 and IPv6 unicast in the route processor . The FIB on each node processes Routing Information Base (RIB) updates, performing route resolution and maintaining FIB tables independently in the route processor . FIB tables on each node can be slightly different.
- Adjacency table—a protocol-independent adjacency information base (AIB)

CEF is a primary IP packet-forwarding database for Cisco IOS XR software. CEF is responsible for the following functions:

- Software switching path
- Maintaining forwarding table and adjacency tables (which are maintained by the AIB) for software and hardware forwarding engines

The following features are supported for CEF on Cisco IOS XR software:

- Bundle interface support
- Multipath support
- Route consistency
- High availability features such as packaging, restartability, and Out of Resource (OOR) handling
- OSPFv2 SPF prefix prioritization

- BGP attributes download

CEF Benefits

- Improved performance—CEF is less CPU-intensive than fast-switching route caching. More CPU processing power can be dedicated to Layer 3 services such as quality of service (QoS) and encryption.
- Scalability—CEF offers full switching capacity at each line card.
- Resilience—CEF offers an unprecedented level of switching consistency and stability in large dynamic networks. In dynamic networks, fast-switched cache entries are frequently invalidated due to routing changes. These changes can cause traffic to be process switched using the routing table, rather than fast switched using the route cache. Because the Forwarding Information Base (FIB) lookup table contains all known routes that exist in the routing table, it eliminates route cache maintenance and the fast-switch or process-switch forwarding scenario. CEF can switch traffic more efficiently than typical demand caching schemes.

The following CEF forwarding tables are maintained in Cisco IOS XR software:

- IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets
- IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets
- MPLS LFD database—Stores MPLS Label table for forwarding MPLS packets

Verifying CEF

To view the details of the IPv4 or IPv6 CEF tables, use the following commands:

- show cef {*ipv4 address* | *ipv6 address*} hardware egress

Displays the IPv4 or IPv6 CEF table. The next hop and forwarding interface are displayed for each prefix. The output of the **show cef** command varies by location.

```
Router# show cef 203.0.1.2 hardware egress
  203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0),
0x0 (0x0)
 Updated Nov 20 13:33:23.557
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
   via 203.0.1.2/32, HundredGigE0/9/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
```

- show cef {ipv4 | ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

```
Router#show cef ipv4 summary
Fri Nov 20 13:50:45.239 UTC

Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 4129
```

```
 56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes
 13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default routes
 56 load sharing elements, 24304 bytes, 1 references
 1 shared load sharing elements, 432 bytes
 55 exclusive load sharing elements, 23872 bytes
 0 route delete cache elements
 13 local route bufs received, 1 remote route bufs received,  0 mix bufs received
 13 local routes, 0 remote routes
 13 total local route updates processed
 0 total remote route updates processed
 0 pkts pre-routed to cust card
 0 pkts pre-routed to rp card
 0 pkts received from core card
 0 CEF route update drops, 0 revisions of existing leaves
 0 CEF route update drops due to version mis-match
 Resolution Timer: 15s
 0 prefixes modified in place
 0 deleted stale prefixes
 0 prefixes with label imposition, 0 prefixes with label information
 0 LISP EID prefixes, 0 merged, via 0 rlocs
28 next hops
 1 incomplete next hop

0 PD backwalks on LDIs with backup path
```

- show cef { ipv4 address | ipv6 address } detail

Displays the details of the IPv4 or IPv6 CEF table.

```
Router#show cef 203.0.1.2 detail
203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0), 0x0
 (0x0)
 Updated Nov 20 13:33:23.556
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
  gateway array (0x8d84beb0) reference count 1, flags 0x0, source aib (10), 0 backups
                [2 type 3 flags 0x8401 (0x8d99a598) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8daeedf0, sh-ldi=0x8d99a598]
  gateway array update type-time 1 Nov 20 13:33:23.556
 LDI Update time Nov 20 13:33:23.556
 LW-LDI-TS Nov 20 13:33:23.556
   via 203.0.1.2/32, HundredGigE0/9/0/0, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
        Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/9/0/0        203.0.1.2
```

- show adjacency detail

Displays detailed adjacency information, including Layer 2 information for each interface. The output of the show adjacency command varies by location.

```
Router#show adjacency detail

------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------
Interface                  Address                 Version  Refcount Protocol
Hu0/9/0/0                 (interface)                  13        1(    0)
```

```
                            (interface entry)
                            mtu: 1500, flags 1 4

Hu0/9/0/1                   (interface)                   31        1(    0)
                            (interface entry)
                            mtu: 1500, flags 1 4
```

Per-Flow Load Balancing

The system inherently supports the 7-tuple hash algorithm. Load balancing describes the functionality in a router that distributes packets across multiple links based on Layer 3 (network layer) and Layer 4 (transport layer) routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.

Per-flow load balancing performs these functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Incoming data traffic is evenly distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows . If it is an IPv4 or an IPv6 payload, then a 7-tuple hashing is done. If it is an MPLS payload with three or less labels, then the hardware parses the payload underneath and identifies whether the payload packet has an IPv4 or an IPv6 header. If it is an IPv4 or IPv6 header, then a 4-tuple hashing is performed based on the IP source, IP destination, router ID, and label stack; otherwise, an MPLS label based hashing is performed. In case of MPLS label-based hashing, the top 4 labels are used in hash computation.
- A 7-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface. In addition, load balancing over member links can occur within a Layer 2 bundle interface.
- The 7-tuple load-balance hash calculation contains:
  - Source IP address
  - Destination IP address
  - IP Protocol type
  - Router ID
  - Source port
  - Destination port
  - Input interface

Load balancing decisions are taken based on a packet header, type of load balancing, type of scenario and platform specifics as follows:

- Packet header can contain one or many MAC, MPLS, IPv4 or IPv6 address, TCPor UDP headers, and so on. Packet header can contain one or many MAC, MPLS, IPv4 or IPv6 address, TCPor UDP headers, and so on.
- Load balancing can be done during ECMP or LAG (Bundle-Ether) forwarding.

- Scenarios can include IP forwarding, IP tunnel forwarding or decapsulation, MPLS forwarding or disaggregation, or Ethernet forwarding.
- The chipset type contains a packet's fields. These fields are considered for load balancing.

The following tables includes detailed list of options, list of scenarios, and headers fields to specify how ECMP or LAG load balancing is done.

Note:

- Only the fields that are highlighted in bold font are used for load balancing hash calculations. For example, for IP forwarding for IPv4 or IPv6 header and L4 (TCP or UDP) header, ECMP or LAG load balancing is done based on:
  - Source and destination IPv4 or IPv6 addresses
  - L4 source and destination ports

Table 1: ECMP or LAG Load Balancing for IP Forwarding

Header 4	Header 3	Header 2	Header 1
		IPv4	ETH
		IPv6	ETH
	L4	**IPv4**	ETH
	L4	**IPv6**	ETH
GTP	**L4**	**IPv4**	ETH
GTP	**L4**	**IPv6**	ETH

Table 2: ECMP or LAG Load Balancing for IP Tunnel Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	**IPv4**	ETH
	L4*	**IPv4**	**IPv4**	ETH
		IPv6	**IPv4**	ETH
		IPv6	**IPv4**	ETH
	IPv4	**MPLS[1..3]**	**IPv4**	ETH
L4	**IPv4**	**MPLS[1..3]**	**IPv4**	ETH
	IPv6	**MPLS[1..3]**	**IPv4**	ETH
L4	**IPv6**	**MPLS[1..3]**	**IPv4**	ETH
IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
		IPv4	**IPv6**	ETH
	L4*	**IPv4**	**IPv6**	ETH
		IPv6	**IPv6**	ETH
	L4*	**IPv6**	**IPv6**	ETH

Note For MPLS packets, up to four labels are used for hash calculation.

Table 3: ECMP or LAG Load Balancing for IP Tunnel Decapsulation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	IPv4	ETH
	L4	**IPv4**	IPv4	ETH
		IPv6	IPv4	ETH
	L4	**IPv6**	IPv4	ETH
	IPv4	**MPLS[1..3]**	IPv4	ETH
L4	**IPv4**	**MPLS[1..3]**	IPv4	ETH
	IPv6	**MPLS[1..3]**	IPv4	ETH
L4	**IPv6**	**MPLS[1..3]**	IPv4	ETH
IPv4	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH
IPv6	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH
MPLS[7..9]	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH

Note For MPLS packets, up to four labels are used for hash calculation.

Table 4: ECMP or LAG Load Balancing for MPLS Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
	IPv4	**ETH**	**MPLS[1..3]**	ETH
	IPv6	**ETH**	**MPLS[1..3]**	ETH
		IPv4	**MPLS[1..3]**	ETH
		IPv6	**MPLS[1..3]**	ETH
	L4*	**IPv4**	**MPLS[1..3]**	ETH
	L4*	**IPv6**	**MPLS[1..3]**	ETH
	IPv4	**IPv4**	**MPLS[1..3]**	ETH
	IPv6	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv4	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv6	**IPv4**	**MPLS[1..3]**	ETH
		MPLS[4..6]	**MPLS[1..3]**	ETH
IPv4	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv4	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv6	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv6**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH

Note For MPLS packets with multiple labels, hash calculation is done based on the first five labels along other headers.

Table 5: ECMP or LAG Load Balancing for MPLS Deaggregation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	MPLS1	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv6	MPLS1	ETH
	L4*	**IPv4**	MPLS1	ETH
	L4*	**IPv6**	MPLS1	ETH
	IPv4	**IPv4**	MPLS1	ETH
	IPv6	**IPv4**	MPLS1	ETH
L4	IPv4	**IPv4**	MPLS1	ETH
L4	IPv6	**IPv4**	MPLS1	ETH
	IPv4	**ETH**	MPLS1	ETH
	IPv6	**ETH**	MPLS1	ETH
L4	**IPv4**	**ETH**	MPLS1	ETH
L4	**IPv6**	**ETH**	MPLS1	ETH

Table 6: ECMP or LAG Load Balancing for Ethernet Forwarding for IPoE Packets

Header 3	Header 2	Header 1
	IPv4	**ETH**
	IPv6	**ETH**
L4*	**IPv4**	**ETH**
L4*	**IPv6**	**ETH**

Table 7: ECMP or LAG Load Balancing Ethernet Forwarding for IPoE Packets with Complex Headers

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4*	**IPv4**	**ETH**
	L4	**IPv4***	**IPv4**	**ETH**
		IPv6*	**IPv4**	**ETH**
	L4	**IPv6***	**IPv4**	**ETH**
	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**
	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**

Header 5	Header 4	Header 3	Header 2	Header 1
IPv4	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
IPv6	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
MPLS[7..9]	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**

Note For MPLS packets with one through three labels, only the first label is used for load balancing along with other headers.

Table 8: ECMP or LAG Load Balancing Ethernet Forwarding for MPLS packets

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	**ETH**
		IPv4*	**MPLS[1..3]**	**ETH**
		IPv6*	**MPLS[1..3]**	**ETH**
	L4	**IPv4***	**MPLS[1..3]**	**ETH**
	L4	**IPv6***	**MPLS[1..3]**	**ETH**
	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
		MPLS[4..6]*	**MPLS[1..3]**	**ETH**
	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv6	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv4	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv6	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**

Note For MPLS packets with multiple labels, hash calculation is done based on first five labels along with other headers.

Per-Destination Load Balancing

Per destination load balancing is used for packets that transit over a recursive MPLS path (for example, learned through BGP 3107). Per-destination load balancing means the router distributes the packets based on the destination of the route. Given two paths to the same network, all packets for destination1 on that network go over the first path, all packets for destination2 on that network go over the second path, and so on. This preserves packet order, with potential unequal usage of the links. If one host receives the majority of the traffic all packets use one link, which leaves bandwidth on other links unused. A larger number of destination addresses leads to more equally used links.

Configuring Static Route

Routers forward packets using either route information from route table entries that you manually configure or the route information that is calculated using dynamic routing algorithms. Static routes, which define explicit paths between two routers, cannot be automatically updated; you must manually reconfigure static routes when network changes occur. Static routes use less bandwidth than dynamic routes. Use static routes where network traffic is predictable and where the network design is simple. You should not use static routes in large, constantly changing networks because static routes cannot react to network changes. Most networks use dynamic routes to communicate between routers but might have one or two static routes configured for special cases. Static routes are also useful for specifying a gateway of last resort (a default router to which all unroutable packets are sent).

Configuration Example

Create a static route between Router A and B over a HundredGigE interface. The destination IP address is 203.0.1.2/32 and the next hop address is 1.0.0.2.


```
Router(config)#router static address-family ipv4 unicast
Router(config-static-afi)#203.0.1.2/32 HundredGigE 0/9/0/0 1.0.0.2
Router(config-static-afi)#commit
```

Running Configuration

```
Router#show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
 203.0.1.2/32 HundredGigE 0/9/0/0 1.0.0.2
 !
!
```

Verification

Verify that the Next Hop Flags fields indicate COMPLETE for accurate functioning of the configuration.

```
Router#show cef 203.0.1.2/32 hardware egress details location 0/0/CPU0
Wed Nov  6 10:09:23.548 UTC
111.0.0.1/32, version 221, attached, internal 0x1000041 0x0 (ptr 0x8b00ea80) [1], 0x0
(0x8afd9768), 0x0 (0x0)
Updated Nov  6 10:08:07.424
Prefix Len 32, traffic index 0, precedence n/a, priority 2
  gateway array (0x8ae4baf0) reference count 1, flags 0x0, source rib (7), 0 backups
                [2 type 3 flags 0x40008441 (0x8af020c0) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8afd9768, sh-ldi=0x8af020c0]
  gateway array update type-time 1 Nov  6 10:08:07.423
LDI Update time Nov  6 10:08:07.423
LW-LDI-TS Nov  6 10:08:07.424
   via tunnel-ip1, 0 dependencies, recursive [flags 0x8]
    path-idx 0 NHID 0x0 [0x8ae0d728 0x0]
    local adjacency
```

Associated Commands

- router static
- show cef

BGP Attributes Download

The BGP Attributes Download feature enables you to display the installed BGP attributes in CEF.

- The **show cef bgp-attribute** command displays the installed BGP attributes in CEF.
- The **show cef bgp-attribute** *attribute-id* command and the **show cef bgp-attribute** *local-attribute-id* command are used to view the specific BGP attributes by attribute ID and local attribute ID.

Verification

```
Router# show cef bgp-attribute
Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 4129
  56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes
  13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default routes
  56 load sharing elements, 24304 bytes, 1 references
  1 shared load sharing elements, 432 bytes
  55 exclusive load sharing elements, 23872 bytes
  0 route delete cache elements
  13 local route bufs received, 1 remote route bufs received,  0 mix bufs received
  13 local routes, 0 remote routes
  13 total local route updates processed
  0 total remote route updates processed
  0 pkts pre-routed to cust card
  0 pkts pre-routed to rp card
  0 pkts received from core card
  0 CEF route update drops, 0 revisions of existing leaves
  0 CEF route update drops due to version mis-match
  Resolution Timer: 15s
  0 prefixes modified in place
```

```
  0 deleted stale prefixes
  0 prefixes with label imposition, 0 prefixes with label information
  0 LISP EID prefixes, 0 merged, via 0 rlocs
 28 next hops
  1 incomplete next hop

 0 PD backwalks on LDIs with backup path


VRF: default

_____________
Table ID: 0xe0000000. Total number of entries: 0
OOR state: GREEN. Number of OOR attributes: 0
```

Associated Commands

- show cef bgp-attribute

Proactive Address Resolution Protocol and Neighbor Discovery

When CEF installs a route for which there is no layer 2 adjacency information, CEF creates an incomplete layer 3 next-hop and programs it on the hardware. Because of this incomplete programming, the first packet will be forwarded to the software forwarding path. The software forwarding in turn strips off the layer 2 header from the packet and forwards it to ARP (Address Resolution Protocol) or ND (Neighbor Discovery) in order to resolve the layer 2 adjacency information. In such a packet, if there is feature specific information present in the layer 2 header, the software forwarding path fails to strip off the layer 2 header completely and thus ARP or ND is unable to resolve the missing layer 2 adjacency information and thereby this results in traffic being dropped.

Proactive ARP and ND feature solves the above problem by ensuring that CEF proactively triggers ARP or ND in order to resolve the missing layer 2 adjacency information, retrying every 15 seconds until the next-hop information is resolved. Thus, when you configure a static route which has an incomplete next-hop information, this feature automatically triggers ARP or ND resolution.

Configuration

```
/* Enter the configuration mode and configure Proactive ARP/ND */
Router# configure
Router(config)# cef proactive-arp-nd enable
Router(config)# commit
```

Running Config

```
Show running-config
cef proactive-arp-nd enable
end
```