



## Configure Multipoint Layer 2 Services

This module provides the conceptual and configuration information for Multipoint Layer 2 Bridging Services, also called Virtual Private LAN Services (VPLS).



**Note** VPLS supports Layer 2 VPN technology and provides transparent multipoint Layer 2 connectivity for customers. This approach enables service providers to host a multitude of new services such as broadcast TV and Layer 2 VPNs.

- [Prerequisites for Implementing Multipoint Layer 2 Services, on page 1](#)
- [Information About Implementing Multipoint Layer 2 Services, on page 1](#)
- [Configuration Examples for Multipoint Layer 2 Services, on page 9](#)
- [GTP Load Balancing, on page 20](#)
- [Flow Aware Transport Pseudowire \(FAT PW\) , on page 22](#)

## Prerequisites for Implementing Multipoint Layer 2 Services

Before configuring Multipoint Layer 2 Services, ensure that these tasks and conditions are met:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.
- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

## Information About Implementing Multipoint Layer 2 Services

To implement Multipoint Layer 2 Services, you must understand these concepts:

## Multipoint Layer 2 Services Overview

Multipoint Layer 2 Services enable geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

Some of the components present in a Multipoint Layer 2 Services network are described in these sections.




---

**Note** Multipoint Layer 2 services are also called as Virtual Private LAN Services.

---

### Bridge Domain

The native bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports (including VFI). Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain. A learned address is aged out. Incoming frames are mapped to a bridge domain, based on either the ingress port or a combination of both an ingress port and a MAC header field.

When the number of bridge domains exceeds 200, to enable clean up and reprogramming, it takes about 120 seconds for unconfiguring L2VPN and rollback.

The following table details the minimum interval required between unconfiguring L2VPN and rollback:

Number of BDs	Minimum interval in seconds
250	180
500	300
750 or greater	600

### Pseudowires

A pseudowire is a point-to-point connection between pairs of PE routers. Its primary function is to emulate services like Ethernet over an underlying core MPLS network through encapsulation into a common MPLS format. By encapsulating services into a common MPLS format, a pseudowire allows carriers to converge their services to an MPLS network.

### Access Pseudowire is not supported over VPLS Bridge Domain

Access PW is not supported over VPLS bridge domain. Only core PW which is configured under VFI is supported.

Configuration Example

```
l2vpn
 bridge group bg1
```

```
bridge-domain l2vpn
 interface TenGigE0/0/0/13.100
 !
 vfi 1
  neighbor 192.0.2.1 pw-id 12345
  pw-class mpls_csr
 !
 !
 !
```

## Virtual Forwarding Instance

VPLS is based on the characteristic of virtual forwarding instance (VFI). A VFI is a virtual bridge port that is capable of performing native bridging functions, such as forwarding, based on the destination MAC address, source MAC address learning and aging, and so forth.

A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

## VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits. The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

After provisioning attachment circuits, neighbor relationships across the MPLS network for this specific instance are established through a set of manual commands identifying the end PEs. When the neighbor association is complete, a full mesh of pseudowires is established among the network-facing provider edge devices, which is a gateway between the MPLS core and the customer domain.

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. This also requires all of the PE routers that are participating in a VPLS instance to form emulated virtual circuits (VCs) among them.

Now, the service provider network starts switching the packets within the bridged domain specific to the customer by looking at destination MAC addresses. All traffic with unknown, broadcast, and multicast destination MAC addresses is flooded to all the connected customer edge devices, which connect to the service provider network. The network-facing provider edge devices learn the source MAC addresses as the packets are flooded. The traffic is unicasted to the customer edge device for all the learned MAC addresses.

## VPLS for Layer 2 Switching

VPLS technology includes the capability of configuring the router to perform Layer 2 bridging. In this mode, the router can be configured to operate like other Cisco switches.

The storm control that is applied to multiple subinterfaces of the same physical port pertains to that physical port only. All subinterfaces with storm control configured are policed as aggregate under a single policer rate shared by all EFPs. None of the subinterfaces are configured with a dedicated policer rate. When a storm

occurs on several subinterfaces simultaneously, and because subinterfaces share the policer, you can slightly increase the policer rate to accommodate additional policing.

These features are supported:

- Bridging IOS XR Trunk Interfaces
- Bridging on EFPs

## Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling

The Cisco IOS Software encodes the NLRI length in the first byte in bits format in the BGP Update message. However, the Cisco IOS XR Software interprets the NLRI length in 2 bytes. Therefore, when the BGP neighbor with VPLS-VPWS address family is configured between the IOS and the IOS XR, NLRI mismatch can happen, leading to flapping between neighbors. To avoid this conflict, IOS supports **prefix-length-size 2** command that needs to be enabled for IOS to work with IOS XR. When the **prefix-length-size 2** command is configured in IOS, the NLRI length is encoded in bytes. This configuration is mandatory for IOS to work with IOS XR.

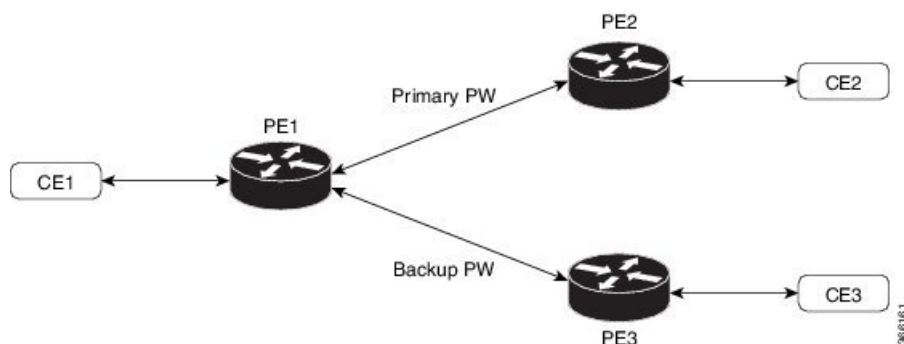
This is a sample IOS configuration with the **prefix-length-size 2** command:

```
router bgp 1
  address-family l2vpn vpls
    neighbor 5.5.5.2 activate
    neighbor 5.5.5.2 prefix-length-size 2 -----> NLRI length = 2 bytes
  exit-address-family
```

## Pseudowire Redundancy

Pseudowire redundancy allows you to configure a backup pseudowire in case the primary pseudowire fails. When the primary pseudowire fails, the PE router can switch to the backup pseudowire. You can elect to have the primary pseudowire resume operation after it becomes functional. The primary pseudowire fails when the PE router fail or due to any network related outage.

**Figure 1: Pseudowire Redundancy**



### Forcing a Manual Switchover to the Backup Pseudowire

To force the router to switch over to the backup or switch back to the primary pseudowire, use the **l2vpn switchover** command in EXEC mode.

A manual switchover is made only if the peer specified in the command is actually available and the cross-connect moves to the fully active state when the command is entered.

## Configuration

This section describes how you can configure pseudowire redundancy.

```

/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 192.168.0.1 pw-id 1
Router(config-subif)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-subif)# commit

/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-subif)# commit

```

## Running Configuration

```

/* On PE1 */
!
l2vpn
xconnect group XCON1
p2p XCON1_P2P2
interface GigabitEthernet 0/1/0/0.1
neighbor ipv4 172.16.0.1 pw-id 1
backup neighbor 192.168.0.1 pw-id 1
!

/* On PE2 */
!
l2vpn
xconnect group XCON1
p2p XCON1_P2P2
interface GigabitEthernet 0/1/0/0.1
neighbor ipv4 10.0.0.1 pw-id 1
!

/* On PE3 */
!
l2vpn
xconnect group XCON1
p2p XCON1_P2P2
interface GigabitEthernet 0/1/0/0.1
neighbor ipv4 10.0.0.1 pw-id 1
!

```

## Verification

Verify that the configured pseudowire redundancy is up.

```
/* On PE1 */
```

```
Router#show l2vpn xconnect group XCON_1
```

```
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
```

XConnect Group	Name	ST	Segment 1 Description	ST	Segment 2 Description	ST
XCON_1	XCON1_P2P2	UP	Gi0/1/0/0.1	UP	172.16.0.1 Backup 192.168.0.1	UP 1000 1000 SB

```
/* On PE2 */
```

```
Router#show l2vpn xconnect group XCON_1
```

```
Tue Jan 17 15:36:12.327 UTC
```

```
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
```

XConnect Group	Name	ST	Segment 1 Description	ST	Segment 2 Description	ST
XCON_1	XCON1_P2P2	UP	BE100.1	UP	10.0.0.1	1000 UP

```
/* On PE3 */
```

```
Router#show l2vpn xconnect group XCON_1
```

```
Tue Jan 17 15:38:04.785 UTC
```

```
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
```

XConnect Group	Name	ST	Segment 1 Description	ST	Segment 2 Description	ST
XCON_1	XCON1_P2P2	DN	BE100.1	UP	10.0.0.1	1000 SB

```
Router#show l2vpn xconnect summary
```

```
Number of groups: 3950
```

```
Number of xconnects: 3950
```

```
Up: 3950 Down: 0 Unresolved: 0 Partially-programmed: 0
```

```
AC-PW: 3950 AC-AC: 0 PW-PW: 0 Monitor-Session-PW: 0
```

```
Number of Admin Down segments: 0
```

```
Number of MP2MP xconnects: 0
```

```
Up 0 Down 0
```

```
Advertised: 0 Non-Advertised: 0
```

```
Number of CE Connections: 0
```

```
Advertised: 0 Non-Advertised: 0
```

```
Backup PW:
```

```
Configured : 3950
```

```
UP : 0
```

```
Down : 0
```

```
Admin Down : 0
```

```
Unresolved : 0
```

```
Standby : 3950
```

```
Standby Ready: 0
```

```
Backup Interface:
  Configured   : 0
  UP           : 0
  Down        : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby     : 0
```

## MAC Address-related Parameters

The MAC address table contains a list of the known MAC addresses and their forwarding information. In the current VPLS design, the MAC address table and its management are maintained on the route processor (RP) card.

These topics provide information about the MAC address-related parameters:

### MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To obtain flooding within VPLS broadcast models, all unknown unicast, broadcast, and multicast frames are flooded over the corresponding pseudowires and to all attachment circuits. Therefore, a PE must replicate packets across both attachment circuits and pseudowires.

### MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with a pseudowire or attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning, which is flooded to all bridge ports.

### MAC Address Source-based Learning

When a frame arrives on a bridge port (for example, pseudowire or attachment circuit) and the source MAC address is unknown to the receiving PE router, the source MAC address is associated with the pseudowire or attachment circuit. Outbound frames to the MAC address are forwarded to the appropriate pseudowire or attachment circuit.

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are propagated and programs the hardware for the router.



---

**Note** Static MAC move is not supported from one port, interface, or AC to another port, interface, or AC. For example, if a static MAC is configured on AC1 (port 1) and then, if you send a packet with the same MAC as source MAC on AC2 (port 2), then you can't attach this MAC to AC2 as a dynamic MAC. Therefore, do not send any packet with a MAC as any of the static MAC addresses configured.

---

The number of learned MAC addresses is limited through configurable per-port and per-bridge domain MAC address limits.

### MAC Address Aging

A MAC address in the MAC table is considered valid only for the duration of the MAC address aging time. When the time expires, the relevant MAC entries are repopulated. When the MAC aging time is configured

only under a bridge domain, all the pseudowires and attachment circuits in the bridge domain use that configured MAC aging time.

A bridge forwards, floods, or drops packets based on the bridge table. The bridge table maintains both static entries and dynamic entries. Static entries are entered by the network manager or by the bridge itself. Dynamic entries are entered by the bridge learning process. A dynamic entry is automatically removed after a specified length of time, known as *aging time*, from the time the entry was created or last updated.

If hosts on a bridged network are likely to move, decrease the aging-time to enable the bridge to adapt to the change quickly. If hosts do not transmit continuously, increase the aging time to record the dynamic entries for a longer time, thus reducing the possibility of flooding when the hosts transmit again.

## MAC Address Limit

The MAC address limit is used to limit the number of learned MAC addresses.

### Configure MAC Address Limit

Configure the MAC address limit using the **maximum** command. The MAC address learning is restricted to the configured limit.

When the number of learned MAC addresses reaches the configured limit, you can configure the bridge behavior by using the **action** command. You can configure the action to perform one of the following:

- **flood**: All the unknown unicast packets, with unknown destinations addresses, are flooded over the bridge.
- **no-flood**: All the unknown unicast packets, with unknown destination addresses, are dropped.
- **shutdown** : All the packets are dropped.

When the MAC limit is exceeded, use the **notification {both | none | trap}** command to send notifications in one of the following forms:

- **trap**: Sends Simple Network Management Protocol (SNMP) trap notification.
- **both**: Sends both syslog and trap notifications.
- **none**: No notifications are sent.

By default, syslog message is sent.




---

**Note** Though you can modify the MAC address limit under the bridge domain, due to hardware limitation, the modification does not take effect.

---

### Configuration Example

In this example, MAC address limit is configured as 5000 and MAC limit action is set to flood the packets. As notification is not configured, syslog entries are sent when the MAC limit is exceeded.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg-0
Router(config-l2vpn-bg)# bridge-domain bd-0
Router(config-l2vpn-bg-bd)# mac
```



```
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 5000
Router(config-l2vpn-bg-bd-mac-limit)# action flood
```

### Verification

Use the **show l2vpn bridge-domain** command to view the MAC address limit configuration.

```
Router# show l2vpn bridge-domain bd-name bd-0 detail
Legend: pp = Partially Programmed.
Bridge group: bg-0, bridge-domain: bd-0, id: 25, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
MAC limit: 5000, Action: flood, Notification: syslog
  MAC limit reached: no, threshold: 80%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
```

## MAC Address Withdrawal

For faster VPLS convergence, you can remove or unlearn the MAC addresses that are learned dynamically. The Label Distribution Protocol (LDP) Address Withdrawal message is sent with the list of MAC addresses, which need to be withdrawn to all other PEs that are participating in the corresponding VPLS service.

For the Cisco IOS XR VPLS implementation, a portion of the dynamically learned MAC addresses are cleared by using the MAC addresses aging mechanism by default. The MAC address withdrawal feature is added through the LDP Address Withdrawal message. To enable the MAC address withdrawal feature, use the **withdrawal** command in l2vpn bridge group bridge domain MAC configuration mode. To verify that the MAC address withdrawal is enabled, use the **show l2vpn bridge-domain** command with the **detail** keyword.




---

**Note** By default, the LDP MAC Withdrawal feature is enabled on Cisco IOS XR.

---

The LDP MAC Withdrawal feature is generated due to these events:

- Attachment circuit goes down. You can remove or add the attachment circuit through the CLI.
- MAC withdrawal messages are received over a VFI pseudowire. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR software supports only a wildcard MAC address withdrawal.

## Configuration Examples for Multipoint Layer 2 Services

This section includes these configuration examples:

## Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example

These configuration examples show how to create a Layer 2 VFI with a full-mesh of participating Multipoint Layer 2 Services provider edge (PE) nodes.

This configuration example shows how to configure PE 1:

```
configure
l2vpn
  bridge group 1
    bridge-domain PE1-VPLS-A
    interface TenGigE0/0/0/0
      vfi 1
        neighbor 172.16.0.1 pw-id 1
        neighbor 192.168.0.1 pw-id 1
      !
    !
  interface loopback 0
    ipv4 address 10.0.0.1 255.0.0.0
```

This configuration example shows how to configure PE 2:

```
configure
l2vpn
  bridge group 1
    bridge-domain PE2-VPLS-A
    interface TenGigE0/0/0/1

    vfi 1
      neighbor 10.0.0.1 pw-id 1
      neighbor 192.168.0.1 pw-id 1
    !
  !
  interface loopback 0
    ipv4 address 172.16.0.1 255.240.0.0
```

This configuration example shows how to configure PE 3:

```
configure
l2vpn
  bridge group 1
    bridge-domain PE3-VPLS-A
    interface TenGigE0/0/0/2
      vfi 1
        neighbor 10.0.0.1 pw-id 1
        neighbor 172.16.0.1 pw-id 1
      !
    !
  interface loopback 0
    ipv4 address 192.168.0.1 255.255.0.0
```

## Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example

This configuration shows how to configure Multipoint Layer 2 Services for a PE-to-CE nodes:

```
configure
interface TenGigE0/0/0/0
  l2transport---AC interface
```

```
no ipv4 address
no ipv4 directed-broadcast
negotiation auto
no cdp enable
```

## Displaying MAC Address Withdrawal Fields: Example

This sample output shows the MAC address withdrawal fields:

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain detail
```

```
Legend: pp = Partially Programmed.
Bridge group: 222, bridge-domain: 222, id: 0, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  Dynamic ARP Inspection: disabled, Logging: disabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 snooping: disabled
  IGMP Snooping: enabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1
  Filter MAC addresses:
  P2MP PW: disabled
  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up)
  List of ACs:
    AC: TenGigE0/2/0/1.7, state is up
      Type VLAN; Num Ranges: 1
      Outer Tag: 21
      VLAN ranges: [22, 22]
      MTU 1508; XC ID 0x208000b; interworking none
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 4000, Action: none, Notification: syslog
      MAC limit reached: no
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 snooping: disabled
      IGMP Snooping: enabled
```

## Displaying MAC Address Withdrawal Fields: Example

```

IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: bridge-domain policer
Static MAC addresses:
Statistics:
  packets: received 714472608 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 97708776
  bytes: received 88594603392 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 12115888224
  MAC move: 0
Storm control drop counters:
  packets: broadcast 0, multicast 0, unknown unicast 0
  bytes: broadcast 0, multicast 0, unknown unicast 0
Dynamic ARP inspection drop counters:
  packets: 0, bytes: 0
IP source guard drop counters:
  packets: 0, bytes: 0
List of VFIs:
VFI 222 (up)
PW: neighbor 10.0.0.1, PW ID 222, state is up ( established )
PW class not set, XC ID 0xc000000a
Encapsulation MPLS, protocol LDP
Source address 21.21.21.21
PW type Ethernet, control word disabled, interworking none
Sequencing not set

PW Status TLV in use
-----
MPLS      Local                               Remote
-----
Label     24017                                     24010
Group ID  0x0                                       0x0
Interface 222                                       222
MTU       1500                                      1500
Control word disabled                       disabled
PW type   Ethernet                                 Ethernet
VCCV CV type 0x2                               0x2
          (LSP ping verification)         (LSP ping verification)
VCCV CC type 0x6                               0x6
          (router alert label)           (router alert label)
          (TTL expiry)                   (TTL expiry)
-----

Incoming Status (PW Status TLV):
  Status code: 0x0 (Up) in Notification message
  MIB cpwVcIndex: 3221225482
  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
  Last time status changed: 01/03/2017 11:21:01 (00:01:43 ago)
  Last time PW went down: 01/03/2017 11:15:21 (00:07:23 ago)
  MAC withdraw messages: sent 0, received 0
  Forward-class: 0
  Static MAC addresses:
  Statistics:
    packets: received 95320440 (unicast 0), sent 425092569
    bytes: received 11819734560 (unicast 0), sent 52711478556
    MAC move: 0
  Storm control drop counters:
    packets: broadcast 0, multicast 0, unknown unicast 0
    bytes: broadcast 0, multicast 0, unknown unicast 0
  DHCPv4 snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  VFI Statistics:
    drops: illegal VLAN 0, illegal length 0

```

## Bridging on IOS XR Trunk Interfaces: Example

This example shows how to configure a Cisco NCS 5000 Series Routers as a simple L2 switch.

### Important notes:

Create a bridge domain that has four attachment circuits (AC). Each AC is an IOS XR trunk interface (i.e. not a subinterface/EFP).

- This example assumes that the running config is empty, and that all the components are created.
- This example provides all the necessary steps to configure the Cisco NCS 5000 Series Routers to perform switching between the interfaces. However, the commands to prepare the interfaces such as no shut, negotiation auto, etc., have been excluded.
- The bridge domain is in a no shut state, immediately after being created.
- Only trunk (i.e. main) interfaces are used in this example.
- The trunk interfaces are capable of handling tagged (i.e. IEEE 802.1Q) or untagged (i.e. no VLAN header) frames.
- The bridge domain learns, floods, and forwards based on MAC address. This functionality works for frames regardless of tag configuration.
- The bridge domain entity spans the entire system. It is not necessary to place all the bridge domain ACs on a single LC. This applies to any bridge domain configuration.
- The show bundle and the show l2vpn bridge-domain commands are used to verify that the router was configured as expected, and that the commands show the status of the new configurations.
- The ACs in this example use interfaces that are in the admin down state.

### Configuration Example

```
RP/0/RSP0/CPU0:router#config
RP/0/RSP0/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/5
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/6
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP0/CPU0:Jul 26 10:48:21.320 EDT: config[65751]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'lab'. Use 'show configuration commit changes 1000000973'
to view the changes.
```

```

RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP0/CPU0:Jul 26 10:48:21.342 EDT: config[65751]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RSP0/CPU0:router#show bundle Bundle-ether10

Bundle-Ether10
  Status:                               Down
  Local links <active/standby/configured>: 0 / 0 / 2
  Local bandwidth <effective/available>: 0 (0) kbps
  MAC address (source):                   0024.f71e.22eb (Chassis pool)
  Minimum active links / bandwidth:       1 / 1 kbps
  Maximum active links:                   64
  Wait while timer:                       2000 ms
  LACP:                                    Operational
    Flap suppression timer:               Off
  mLACP:                                   Not configured
  IPv4 BFD:                               Not configured

Port          Device          State          Port ID          B/W, kbps
-----
Gi0/2/0/5     Local           Configured     0x8000, 0x0001  1000000
  Link is down
Gi0/2/0/6     Local           Configured     0x8000, 0x0002  1000000
  Link is down

RP/0/RSP0/CPU0:router#
RP/0/RSP0/CPU0:router#show l2vpn bridge-domain group examples
Bridge group: examples, bridge-domain: test-switch, id: 2000, state: up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
Filter MAC addresses: 0
ACs: 4 (1 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
List of ACs:
  BE10, state: down, Static MAC addresses: 0
  Gi0/2/0/0, state: up, Static MAC addresses: 0
  Gi0/2/0/1, state: down, Static MAC addresses: 0
  Te0/5/0/1, state: down, Static MAC addresses: 0
List of VFIs:
RP/0/RSP0/CPU0:router#

```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

## SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **l2transport**
4. **interface GigabitEthernet0/2/0/5**
5. **bundle id 10 mode active**
6. **interface GigabitEthernet0/2/0/6**
7. **bundle id 10 mode active**
8. **interface GigabitEthernet0/2/0/0**
9. **l2transport**
10. **interface GigabitEthernet0/2/0/1**
11. **l2transport**
12. **interface TenGigE0/1/0/2**
13. **l2transport**
14. **l2vpn**

15. **bridge group examples**
16. **bridge-domain test-switch**
17. **interface Bundle-ether10**
18. **exit**
19. **interface GigabitEthernet0/2/0/0**
20. **exit**
21. **interface GigabitEthernet0/2/0/1**
22. **exit**
23. **interface TenGigE0/1/0/2**
24. Use the **commit** or **end** command.

## DETAILED STEPS

---

- |                |   |
|----------------|---|
| <b>Step 1</b>  | <b>configure</b><br>Enters global configuration mode.   |
| <b>Step 2</b>  | <b>interface Bundle-ether10</b><br>Creates a new bundle trunk interface.  |
| <b>Step 3</b>  | <b>l2transport</b><br>Changes Bundle-ether10 from an L3 interface to an L2 interface.   |
| <b>Step 4</b>  | <b>interface GigabitEthernet0/2/0/5</b><br>Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/5.                |
| <b>Step 5</b>  | <b>bundle id 10 mode active</b><br>Establishes GigabitEthernet0/2/0/5 as a member of Bundle-ether10. The <b>mode active</b> keywords specify LACP protocol. |
| <b>Step 6</b>  | <b>interface GigabitEthernet0/2/0/6</b><br>Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/6.                |
| <b>Step 7</b>  | <b>bundle id 10 mode active</b><br>Establishes GigabitEthernet0/2/0/6 as a member of Bundle-ether10. The <b>mode active</b> keywords specify LACP protocol. |
| <b>Step 8</b>  | <b>interface GigabitEthernet0/2/0/0</b><br>Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/0.                |
| <b>Step 9</b>  | <b>l2transport</b><br>Change GigabitEthernet0/2/0/0 from an L3 interface to an L2 interface.  |
| <b>Step 10</b> | <b>interface GigabitEthernet0/2/0/1</b><br>Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/1.                |
| <b>Step 11</b> | <b>l2transport</b><br>Change GigabitEthernet0/2/0/1 from an L3 interface to an L2 interface.  |

- Step 12**      **interface TenGigE0/1/0/2**  
Enters interface configuration mode. Changes configuration mode to act on TenGigE0/1/0/2.
- Step 13**      **l2transport**  
Changes TenGigE0/1/0/2 from an L3 interface to an L2 interface.
- Step 14**      **l2vpn**  
Enters L2VPN configuration mode.
- Step 15**      **bridge group examples**  
Creates the bridge group **examples**.
- Step 16**      **bridge-domain test-switch**  
Creates the bridge domain **test-switch**, that is a member of bridge group **examples**.
- Step 17**      **interface Bundle-ether10**  
Establishes Bundle-ether10 as an AC of bridge domain test-switch.
- Step 18**      **exit**  
Exits bridge domain AC configuration submode, allowing next AC to be configured.
- Step 19**      **interface GigabitEthernet0/2/0/0**  
Establishes GigabitEthernet0/2/0/0 as an AC of bridge domain **test-switch**.
- Step 20**      **exit**  
Exits bridge domain AC configuration submode, allowing next AC to be configured.
- Step 21**      **interface GigabitEthernet0/2/0/1**  
Establishes GigabitEthernet0/2/0/1 as an AC of bridge domain **test-switch**.
- Step 22**      **exit**  
Exits bridge domain AC configuration submode, allowing next AC to be configured.
- Step 23**      **interface TenGigE0/1/0/2**  
Establishes interface TenGigE0/1/0/2 as an AC of bridge domain **test-switch**.
- Step 24**      Use the **commit** or **end** command.  
**commit** - Saves the configuration changes and remains within the configuration session.  
**end** - Prompts user to take one of these actions:
- **Yes** - Saves configuration changes and exits the configuration session.
  - **No** - Exits the configuration session without committing the configuration changes.
  - **Cancel** - Remains in the configuration mode, without committing the configuration changes.
-



## Bridging on Ethernet Flow Points: Example

This example shows how to configure a Cisco NCS 5000 Series Router to perform Layer 2 switching on traffic that passes through Ethernet Flow Points (EFPs). EFP traffic typically has one or more VLAN headers. Although both IOS XR trunks and IOS XR EFPs can be combined as attachment circuits in bridge domains, this example uses EFPs exclusively.

### Important notes:

- An EFP is a Layer 2 subinterface. It is always created under a trunk interface. The trunk interface must exist before the EFP is created.
- In an empty configuration, the bundle interface trunk does not exist, but the physical trunk interfaces are automatically configured. Therefore, only the bundle trunk is created.
- In this example the subinterface number and the VLAN IDs are identical, but this is out of convenience, and is not a necessity. They do not need to be the same values.
- The bridge domain test-efp has three attachment circuits (ACs). All the ACs are EFPs.
- Only frames with a VLAN ID of 999 enter the EFPs. This ensures that all the traffic in this bridge domain has the same VLAN encapsulation.
- The ACs in this example use interfaces that are in the admin down state (**unresolved** state). Bridge domains that use nonexistent interfaces as ACs are legal, and the commit for such configurations does not fail. In this case, the status of the bridge domain shows **unresolved** until you configure the missing interface.

### Configuration Example

```
RP/0/RSP1/CPU0:router#configure
RP/0/RSP1/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP1/CPU0:router(config-if)#interface Bundle-ether10.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface GigabitEthernet0/6/0/5
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/6
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/7.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface TenGigE0/1/0/2.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#l2vpn
RP/0/RSP1/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP1/CPU0:router(config-l2vpn-bg)#bridge-domain test-efp
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/6/0/7.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP1/CPU0:router#
RP/0/RSP1/CPU0:router#show l2vpn bridge group examples
Fri Jul 23 21:56:34.473 UTC Bridge group: examples, bridge-domain: test-efp, id: 0, state:
up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
Filter MAC addresses: 0
ACs: 3 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
List of ACs:
```

```

BE10.999, state: down, Static MAC addresses: 0
Gi0/6/0/7.999, state: unresolved, Static MAC addresses: 0
Te0/1/0/2.999, state: down, Static MAC addresses: 0
List of VFIs:
RP/0/RSP1/CPU0:router#

```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

## SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **interface Bundle-ether10.999 l2transport**
4. **encapsulation dot1q 999**
5. **interface GigabitEthernet0/6/0/5**
6. **bundle id 10 mode active**
7. **interface GigabitEthernet0/6/0/6**
8. **bundle id 10 mode active**
9. **interface GigabitEthernet0/6/0/7.999 l2transport**
10. **encapsulation dot1q 999**
11. **interface TenGigE0/1/0/2.999 l2transport**
12. **encapsulation dot1q 999**
13. **l2vpn**
14. **bridge group examples**
15. **bridge-domain test-efp**
16. **interface Bundle-ether10.999**
17. **exit**
18. **interface GigabitEthernet0/6/0/7.999**
19. **exit**
20. **interface TenGigE0/1/0/2.999**
21. Use the **commit** or **end** command.

## DETAILED STEPS

- 
- |               |   |
|---------------|---|
| <b>Step 1</b> | <b>configure</b><br>Enters global configuration mode.   |
| <b>Step 2</b> | <b>interface Bundle-ether10</b><br>Creates a new bundle trunk interface.                      |
| <b>Step 3</b> | <b>interface Bundle-ether10.999 l2transport</b><br>Creates an EFP under the new bundle trunk. |
| <b>Step 4</b> | <b>encapsulation dot1q 999</b><br>Assigns VLAN ID of 999 to this EFP.                         |
| <b>Step 5</b> | <b>interface GigabitEthernet0/6/0/5</b>   |

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/5.

**Step 6**      **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 7**      **interface GigabitEthernet0/6/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/6.

**Step 8**      **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 9**      **interface GigabitEthernet0/6/0/7.999 l2transport**

Creates an EFP under GigabitEthernet0/6/0/7.

**Step 10**     **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

**Step 11**     **interface TenGigE0/1/0/2.999 l2transport**

Creates an EFP under TenGigE0/1/0/2.

**Step 12**     **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

**Step 13**     **l2vpn**

Enters L2VPN configuration mode.

**Step 14**     **bridge group examples**

Creates the bridge group named **examples**.

**Step 15**     **bridge-domain test-efp**

Creates the bridge domain named **test-efp**, that is a member of bridge group **examples**.

**Step 16**     **interface Bundle-ether10.999**

Establishes Bundle-ether10.999 as an AC of the bridge domain named **test-efp**.

**Step 17**     **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 18**     **interface GigabitEthernet0/6/0/7.999**

Establishes GigabitEthernet0/6/0/7.999 as an AC of the bridge domain named **test-efp**.

**Step 19**     **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 20**     **interface TenGigE0/1/0/2.999**

Establishes interface TenGigE0/1/0/2.999 as an AC of bridge domain named **test-efp**.

**Step 21**     Use the **commit** or **end** command.

**commit** - Saves the configuration changes and remains within the configuration session.

**end** - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

## GTP Load Balancing

*Table 1: Feature History Table*

Feature Name	Release Information	Feature Description
GTP Load Balancing	Release 7.3.2	In addition to the source IP address, destination IP address, and port number, this functionality enables using the unique tunnel endpoint identifier (TEID) to compute load balancing (or hashing) of traffic in tunnels between ports. Using the TEID ensures that load balancing occurs even if the other parameters don't have unique values, thus enabling efficient use of bandwidth and providing a reliable network.  This functionality introduces the <b>hw-module loadbalancing gtp enable</b> command.

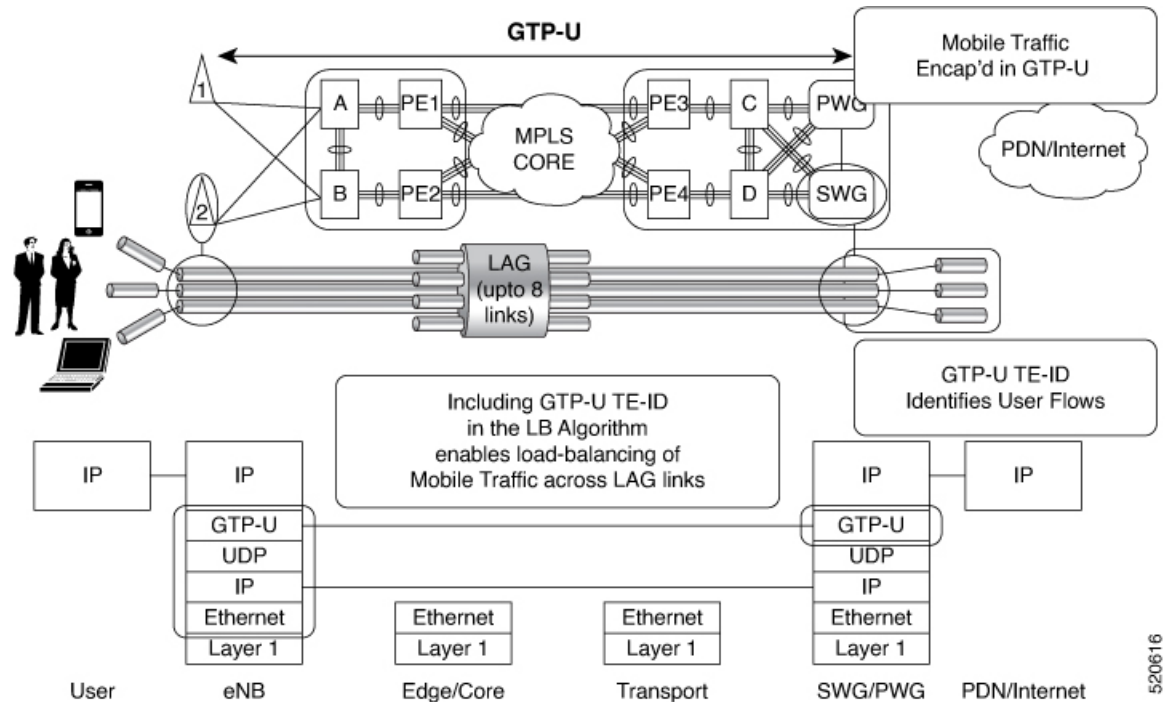
### What is GTP?

GTP is a tunnel control and management protocol among General Packet Radio Service (GPRS) support nodes. Wireless networks use GTP tunnels to deliver mobile data. GTP includes GTP signaling (GTP-C) and data transfer (GTP-U) procedures. GTP-C specifies a tunnel control and management protocol, and is used to create, delete, and modify tunnels. GTP-U uses a tunneling mechanism to provide a service for carrying user data packets over the network.

### What is GTP Load Balancing?

The following figure shows an illustration of the mobile transport GTP-U load balancing.

Figure 2: Mobile Transport GTP-U Load-Balancing



The global L3 flow-based load balancing considers the following fields:

- source address
- destination address
- router ID
- source port
- destination port

However, for GTP traffic, there are a limited number of unique values for these fields; this causes an uneven distribution of traffic. Sometimes, to facilitate redundancy and load balancing in a network, equal-cost paths exist to different destinations. Load balancing doesn't occur in such scenarios as the source and destination IP addresses and L4 ports are the same.

To achieve a greater distribution of traffic over equal-cost links, you can enable GTP TEID (Tunnel Endpoint ID) in the hash computation algorithm using the **hw-module loadbalancing gtp enable** command. Doing so ensures that the load balancing (hashing) computation algorithm includes the GTP TEID, unique for each traffic flow. The GTP load-balancing feature allows efficient distribution of traffic in mobile networks and provides increased reliability and availability for the network.

If the packet is UDP and the destination port is the GTP-U port (port number 2152), the GTP TEID is considered for loadbalancing.

The TEID in the GTP header of a GTP packet identifies individual tunnel endpoints, thus achieving better mobile traffic load balancing within any given GRE tunnel. Also, this helps in load-balancing GTP traffic over bundles at transit routers.

If TEID is present, load balancing based on tunnel endpoints is supported for Version 1 GTP packet and GTP version 2. For GTP version 0, load balancing occurs only if the fields described earlier have unique values, because there's no TEID in version 0.



**Note** GTP load balancing is performed only for GTP-U (user data) packets. The GTP-C (control data) packets use a different destination port number of 2123 and hence, are subject to only the global L3 flow-based load balancing.

### GTP Load Balancing Guidelines and Limitations

- GTP load balancing is performed on IPv4 or IPv6 incoming packets with GTP payloads and on MPLS incoming labeled packets.
- This feature supports GTP hashing only when the GTP UDP port is 2152.
- The number of MPLS label stacks in the transport layer is limited to three for GTP hashing. GTP hashing isn't considered when the MPLS label stack exceeds three.
- You need not reboot the router after configuring or unconfiguring the **hw-module loadbalancing gtp enable** for GTP load balancing over MPLS to take effect.

### Configuration

To enable GTP load balancing, configure the **hw-module loadbalancing gtp enable** command.

```
Router#config
Router(config)#hw-module loadbalancing gtp enable
Router(config)#commit
Router(config)#end
```

### Verification

Run the following command to verify that GTP load balancing is enabled.

```
Router#show prm drv diagshell "getreg ING_GTP_CONTROL"
ING_GTP_CONTROL.ipipe0[1][0x3a004f00]=1: <GTP_HDR_FIRST_BYTE_MASK=0,
GTP_HDR_FIRST_BYTE=0,GTP_ENABLE=1>
```

### Associated Commands

[hw-module loadbalancing gtp enable](#)

## Flow Aware Transport Pseudowire (FAT PW)

Routers typically loadbalance traffic based on the lower most label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric loadbalancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) to a destination PE.

Flow-Aware Transport Pseudowires (FAT PW) provide the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to loadbalance traffic. FAT PWs are used to loadbalance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on

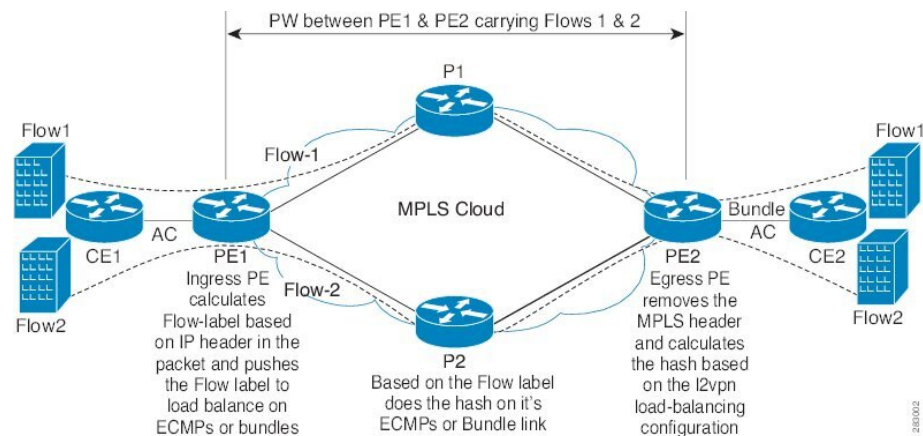
indivisible packet flows entering a pseudowire; and is inserted as the lower most label in the packet. Routers can use the flow label for loadbalancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.



**Note** Based on the number of traffic flows, a SMAC based hashing tuple creates the same flow label or less number of flow labels.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

**Figure 3: FAT PW with two flows distributing over ECMPs and Bundle-Links**



An additional label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set and inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it's possible to distribute flows over ECMPs and link bundles.

You cannot send MPLS OAM ping traffic over a FAT PW, since there is no flow label support for MPLS OAM.

