

Implementing MPLS Traffic Engineering

Traditional IP routing emphasizes on forwarding traffic to the destination as fast as possible. As a result, the routing protocols find out the least-cost route according to its metric to each destination in the network and every router forwards the packet based on the destination IP address and packets are forwarded hop-by-hop. Thus, traditional IP routing does not consider the available bandwidth of the link. This can cause some links to be over-utilized compared to others and bandwidth is not efficiently utilized. Traffic Engineering (TE) is used when the problems result from inefficient mapping of traffic streams onto the network resources. Traffic engineering allows you to control the path that data packets follow and moves traffic flows from congested links to non-congested links that would not be possible by the automatically computed destination-based shortest path.

Multiprotocol Label Switching (MPLS) with its label switching capabilities, eliminates the need for an IP route look-up and creates a virtual circuit (VC) switching function, allowing enterprises the same performance on their IP-based network services as with those delivered over traditional networks such as Frame Relay or Asynchronous Transfer Mode (ATM). MPLS traffic engineering (MPLS-TE) relies on the MPLS backbone to replicate and expand upon the TE capabilities of Layer 2 ATM and Frame Relay networks.

MPLS-TE learns the topology and resources available in a network and then maps traffic flows to particular paths based on resource requirements and network resources such as bandwidth. MPLS-TE builds a unidirectional tunnel from a source to a destination in the form of a label switched path (LSP), which is then used to forward traffic. The point where the tunnel begins is called the tunnel headend or tunnel source, and the node where the tunnel ends is called the tunnel tailend or tunnel destination. A router through which the tunnel passes is called the mid-point of the tunnel.

MPLS uses extensions to a link-state based Interior Gateway Protocol (IGP), such as Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF). MPLS calculates TE tunnels at the LSP head based on required and available resources (constraint-based routing). If configured, the IGP automatically routes the traffic onto these LSPs. Typically, a packet that crosses the MPLS-TE backbone travels on a single LSP that connects the ingress point to the egress point. MPLS TE automatically establishes and maintains the LSPs across the MPLS network by using the Resource Reservation Protocol (RSVP).

Note Combination of unlabelled paths protected by labelled paths is not supported.

- Overview of MPLS-TE Features, on page 2
- How MPLS-TE Works, on page 3
- Configuring MPLS-TE, on page 4
- Regular expressions in MPLS-TE tunnel names, on page 32

• MPLS-TE Features - Details, on page 35
• Configuring Performance Measurement, on page 44

Overview of MPLS-TE Features

In MPLS traffic engineering, IGP extensions flood the TE information across the network. Once the IGP distributes the link attributes and bandwidth information, the headend router calculates the best path from head to tail for the MPLS-TE tunnel. This path can also be configured explicitly. Once the path is calculated, RSVP-TE is used to set up the TE LSP (Labeled Switch Path).

To forward the traffic, you can configure autoroute, forward adjacency, or static routing. The autoroute feature announces the routes assigned by the tailend router and its downstream routes to the routing table of the headend router and the tunnel is considered as a directly connected link to the tunnel.

If forward adjacency is enabled, MPLS-TE tunnel is advertised as a link in an IGP network with the link's cost associated with it. Routers outside of the TE domain can see the TE tunnel and use it to compute the shortest path for routing traffic throughout the network.

MPLS-TE provides protection mechanism known as fast reroute to minimize packet loss during a failure. For fast reroute, you need to create back up tunnels. The autotunnel backup feature enables a router to dynamically build backup tunnels when they are needed instead of pre-configuring each backup tunnel and then assign the backup tunnel to the protected interfaces.

DiffServ Aware Traffic Engineering (DS-TE) enables you to configure multiple bandwidth constraints on an MPLS-enabled interface to support various classes of service (CoS). These bandwidth constraints can be treated differently based on the requirement for the traffic class using that constraint.

The MPLS traffic engineering autotunnel mesh feature allows you to set up full mesh of TE tunnels automatically with a minimal set of MPLS traffic engineering configurations. The MPLS-TE auto bandwidth feature allows you to automatically adjusts bandwidth based on traffic patterns without traffic disruption.

The MPLS-TE interarea tunneling feature allows you to establish TE tunnels spanning multiple Interior Gateway Protocol (IGP) areas and levels, thus eliminating the requirement that headend and tailend routers should reside in a single area.

For detailed information about MPLS-TE features, see MPLS-TE Features - Details, on page 35.

Note MPLS-TE Nonstop Routing (NSR) is enabled by default without any user configuration and cannot be disabled.

MPLS-TE NSR means the application is in hot-standby mode and standby MPLS-TE instance is ready to take over from the active instance quickly on RP failover.

Note that the MPLS-TE does not do routing. If there is standby card available then the MPLS-TE instance is in a hot-standby position.

The following output shows the status of MPLS-TE NSR:

```
Router#show mpls traffic-eng nsr status

TE Process Role        : V1 Active
Current Status       : Ready
    Ready since        : Tue Nov 01 10:42:34 UTC 2022 (1w3d ago)
   IDT started         : Tue Nov 01 03:28:48 UTC 2022 (1w3d ago)
    IDT ended          : Tue Nov 01 03:28:48 UTC 2022 (1w3d ago)
Previous Status        : Not ready
    Not ready reason   : Collaborator disconnected
    Not ready since    : Tue Nov 01 10:42:34 UTC 2022 (1w3d ago)
```

During any issues with the MPLS-TE, the NSR on the router gets affected which is displayed in the show redundancy output as follows:

```
Router#show mpls traffic-eng nsr status details
.
.
.

Current active rmf state: 4 (I_READY)
All standby not-ready bits clear - standby should be ready

Current active rmf state for NSR: Not ready
<jid> <node> <name> Reason for standby not NSR-ready
1082 0/RP0/CPU0 te_control TE NSR session not synchronized
Not ready set Wed Nov 19 17:28:14 2022: 5 hours, 23 minutes ago
1082 0/RP1/CPU0 te_control Standby not connected
Not ready set Wed Nov 19 17:29:11 2022: 5 hours, 22 minutes ago
```

Note If you have disabled MPLS-TE NSR and configured only MPLS-TE nonstop forwarding (NSF), performing an RP failover results in a verification error causing unprotected LSPs to flap and triggering FRR on protected LSPs. To avoid the verification error, you must block the path verification using the **mpls traffic-eng topology block-verify-after-rpfo** command.

How MPLS-TE Works

MPLS-TE automatically establishes and maintains label switched paths (LSPs) across the backbone by using RSVP. The path that an LSP uses is determined by the LSP resource requirements and network resources, such as bandwidth. Available resources are flooded by extensions to a link state based Interior Gateway Protocol (IGP). MPLS-TE tunnels are calculated at the LSP headend router, based on a fit between the required and available resources (constraint-based routing). The IGP automatically routes the traffic to these LSPs.

Typically, a packet crossing the MPLS-TE backbone travels on a single LSP that connects the ingress point to the egress point.

The following sections describe the components of MPLS-TE:

Tunnel Interfaces

From a Layer 2 standpoint, an MPLS tunnel interface represents the headend of an LSP. It is configured with a set of resource requirements, such as bandwidth and media requirements, and priority. From a Layer 3 standpoint, an LSP tunnel interface is the headend of a unidirectional virtual link to the tunnel destination.

MPLS-TE Path Calculation Module

This calculation module operates at the LSP headend. The module determines a path to use for an LSP. The path calculation uses a link-state database containing flooded topology and resource information.

RSVP with TE Extensions

RSVP operates at each LSP hop and is used to signal and maintain LSPs based on the calculated path.

MPLS-TE Link Management Module

This module operates at each LSP hop, performs link call admission on the RSVP signaling messages, and keep track on topology and resource information to be flooded.

Link-state IGP

Either Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF) can be used as IGPs. These IGPs are used to globally flood topology and resource information from the link management module.

Label Switching Forwarding

This forwarding mechanism provides routers with a Layer 2-like ability to direct traffic across multiple hops of the LSP established by RSVP signaling.

Configuring MPLS-TE

MPLS-TE requires co-ordination among several global neighbor routers. RSVP, MPLS-TE and IGP are configured on all routers and interfaces in the MPLS traffic engineering network. Explicit path and TE tunnel interfaces are configured only on the head-end routers. MPLS-TE requires some basic configuration tasks explained in this section.

Restrictions for MPLS-TE

- Dark bandwidth is not supported.

Building MPLS-TE Topology

Building MPLS-TE topology, sets up the environment for creating MPLS-TE tunnels. This procedure includes the basic node and interface configuration for enabling MPLS-TE. To perform constraint-based routing, you need to enable OSPF or IS-IS as IGP extension.

Before You Begin

Before you start to build the MPLS-TE topology, the following pre-requisites are required:

- Stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.
- Enable RSVP on the port interface.

Example

This example enables MPLS-TE on a node and then specifies the interface that is part of the MPLS-TE. Here, OSPF is used as the IGP extension protocol for information distribution.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface hundredGigE0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config)# router ospf area 1
RP/0/RP0/CPU0:router(config-ospf)# area 0
RP/0/RP0/CPU0:router(config-ospf-ar)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-ospf-ar)# interface hundredGigE0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit
RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id 192.168.70.1
RP/0/RP0/CPU0:router(config)# commit
```

Example

This example enables MPLS-TE on a node and then specifies the interface that is part of the MPLS-TE. Here, IS-IS is used as the IGP extension protocol for information distribution.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface hundredGigE0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# net 47.0001.0000.0000.0002.00
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide
RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng level 1
RP/0/RP0/CPU0:router(config-isis-af)# exit
RP/0/RP0/CPU0:router(config-isis)# interface hundredGigE0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-isis-if)# exit
RP/0/RP0/CPU0:router(config)# commit
```

Related Topics

- How MPLS-TE Works, on page 3
- Creating an MPLS-TE Tunnel, on page 6

Creating an MPLS-TE Tunnel

Creating an MPLS-TE tunnel is a process of customizing the traffic engineering to fit your network topology. The MPLS-TE tunnel is created at the headend router. You need to specify the destination and path of the TE LSP.

To steer traffic through the tunnel, you can use the following ways:

- Static Routing
- Autoroute Announce
- Forwarding Adjacency

From the 7.1.1 release, IS-IS autoroute announce function is enhanced to redirect traffic from a source IP address prefix to a matching IP address assigned to an MPLS-TE tunnel destination interface.

Note Configuring Segment Routing and Autoroute Destination together is not supported. If autoroute functionality is required in an Segment Routing network, we recommend you to configure Autoroute Announce.

Before You Begin

The following prerequisites are required to create an MPLS-TE tunnel:

- You must have a router ID for the neighboring router.
- Stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

Configuration Example

This example configures an MPLS-TE tunnel on the headend router with a destination IP address 192.168.92.125. The bandwidth for the tunnel, path-option, and forwarding parameters of the tunnel are also configured. You can use static routing, autoroute announce or forwarding adjacency to steer traffic through the tunnel.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface tunnel-te 1
RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125
RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered Loopback0
RP/0/RP0/CPU0:router(config-if)# path-option 1 dynamic
RP/0/RP0/CPU0:router(config-if)# autoroute announce  or forwarding adjacency
RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 100
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Verify the configuration of MPLS-TE tunnel using the following command.

```
RP/0/RP0/CPU0:router# show mpls traffic-engineering tunnels brief

            Signalling Summary:
                LSP Tunnels Process:  running
                       RSVP Process:  running
```

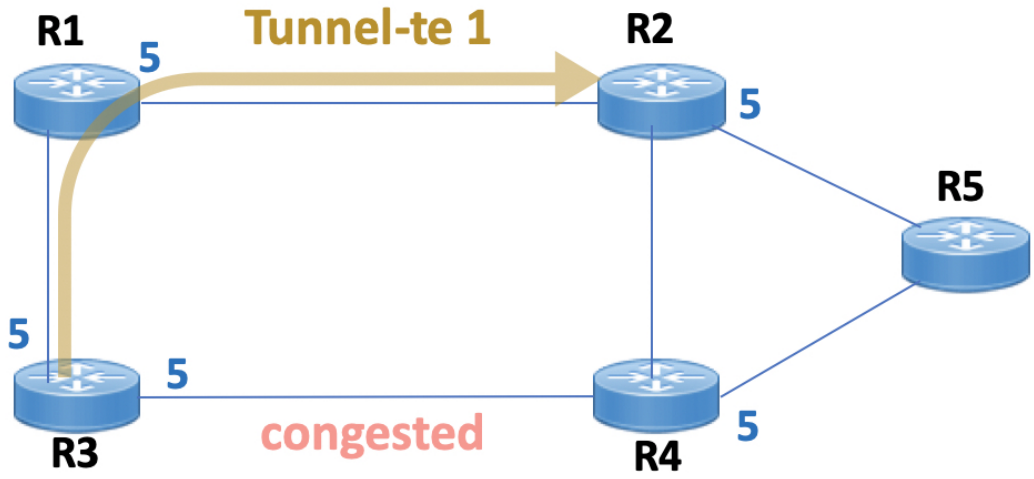
```
             Forwarding:  enabled
Periodic reoptimization:  every 3600 seconds, next in 2538 seconds
 Periodic FRR Promotion:  every 300 seconds, next in 38 seconds
Auto-bw enabled tunnels:  0 (disabled)
           TUNNEL NAME          DESTINATION      STATUS  STATE


          tunnel-te1       192.168.92.125      up       up
Displayed 1 up, 0 down, 0 recovering, 0 recovered heads
```

Automatic Modification Of An MPLS-TE Tunnel's Metric

If the IGP calculation on a router results in an equal cost multipath (ECMP) scenario where next-hop interfaces are a mix of MPLS-TE tunnels and physical interfaces, you may want to ensure that a TE tunnel is preferred. Consider this topology:

Figure 1: MPLS-TE Tunnel



1. All links in the network have a metric of 5.
2. To offload a congested link between R3 and R4, an MPLS-TE tunnel is created from R3 to R2.
3. If the metric of the tunnel is also 5, traffic from R3 to R5 is load-balanced between the tunnel and the physical R3-R4 link.

To ensure that the MPLS-TE tunnel is preferred in such scenarios, configure the **autoroute metric** command on the tunnel interface. The modified metric is applied in the routing information base (RIB), and the tunnel is preferred over the physical path of the same metric. Sample configuration:

```
Router# configure
Router(config)# interface tunnel-te 1
Router(config-if)# autoroute metric relative -1
```

The **autoroute metric** command syntax is **autoroute metric** {**absolute**|**relative**} *value*

- **absolute** enables the absolute metric mode, for a metric range between 1 and 2147483647.
- **relative** enables the relative metric mode, for a metric range between -10 and 10, including zero.

Note Since the **relative** metric is not saved in the IGP database, the advertised metric of the MPLS-TE tunnel remains 5, and doesn't affect SPF calculation outcomes on other nodes.

Related Topics

- How MPLS-TE Works, on page 3
- Building MPLS-TE Topology , on page 5

Configuring Fast Reroute

Fast reroute (FRR) provides link protection to LSPs enabling the traffic carried by LSPs that encounter a failed link to be rerouted around the failure. The reroute decision is controlled locally by the router connected to the failed link. The headend router on the tunnel is notified of the link failure through IGP or through RSVP. When it is notified of a link failure, the headend router attempts to establish a new LSP that bypasses the failure. This provides a path to reestablish links that fail, providing protection to data transfer. The path of the backup tunnel can be an IP explicit path, a dynamically calculated path, or a semi-dynamic path. For detailed conceptual information on fast reroute, see MPLS-TE Features - Details, on page 35

Before You Begin

The following prerequisites are required to create an MPLS-TE tunnel:

- You must have a router ID for the neighboring router.
- Stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

Configuration Example

This example configures fast reroute on an MPLS-TE tunnel. Here, tunnel-te 2 is configured as the back-up tunnel. You can use the **protected-by** command to configure path protection for an explicit path that is protected by another path.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# interface tunnel-te 1
RP/0/RP0/CPU0:router(config-if)# fast-reroute
RP/0/RP0/CPU0:router(config-if)# exit
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface HundredGigabitEthernet0/0/0/3
RP/0/RP0/CPU0:router(config-mpls-te-if)# backup-path tunnel-te 2
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# backup-bw global-pool 5000
RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered Loopback0
RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125
RP/0/RP0/CPU0:router(config-if)# path-option l explicit name backup-path protected by 10
RP/0/RP0/CPU0:router(config-if)# path-option l0 dynamic
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Use the **show mpls traffic-eng fast-reroute database** command to verify the fast reroute configuration.

```
RP/0/RP0/CPU0:router# show mpls traffic-eng fast-reroute database

Tunnel head FRR information:
Tunnel      Out intf/label                      FRR intf/label      Status
----------  ----------------                    ----------------    -------
tt4000       HundredGigabitEthernet 0/0/0/3:34     tt1000:34        Ready
tt4001       HundredGigabitEthernet 0/0/0/3:35     tt1001:35        Ready
tt4002       HundredGigabitEthernet 0/0/0/3:36     tt1001:36        Ready
```

Related Topics

- Configuring MPLS-TE, on page 4
- Configuring Auto-Tunnel Backup, on page 9
- Configuring Next Hop Backup Tunnel, on page 10
- MPLS-TE Features - Details, on page 35

Configuring Auto-Tunnel Backup

The MPLS Traffic Engineering Auto-Tunnel Backup feature enables a router to dynamically build backup tunnels on the interfaces that are configured with MPLS TE tunnels instead of building MPLS-TE tunnels statically.

The MPLS-TE Auto-Tunnel Backup feature has these benefits:

- Backup tunnels are built automatically, eliminating the need for users to pre-configure each backup tunnel and then assign the backup tunnel to the protected interface.
- Protection is expanded—FRR does not protect IP traffic that is not using the TE tunnel or Label Distribution Protocol (LDP) labels that are not using the TE tunnel.

The TE attribute-set template that specifies a set of TE tunnel attributes, is locally configured at the headend of auto-tunnels. The control plane triggers the automatic provisioning of a corresponding TE tunnel, whose characteristics are specified in the respective attribute-set.

Configuration Example

This example configures Auto-Tunnel backup on an interface and specifies the attribute-set template for the auto tunnels. In this example, unused backup tunnels are removed every 20 minutes using a timer and also the range of tunnel interface numbers are specified.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface HundredGigabitEthernet0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-mpls-te-if)# auto-tunnel backup
RP/0/RP0/CPU0:router(config-mpls-te-if-auto-backup)# attribute-set ab
RP/0/RP0/CPU0:router(config-mpls-te)# auto-tunnel backup timers removal unused 20
RP/0/RP0/CPU0:router(config-mpls-te)# auto-tunnel backup tunnel-id min 6000 max 6500
RP/0/RP0/CPU0:router(config)# commit
```

Verification

This example shows a sample output for automatic backup tunnel configuration.

```
RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels brief

            TUNNEL NAME          DESTINATION      STATUS  STATE
             tunnel-te0            200.0.0.3          up  up
             tunnel-te1            200.0.0.3          up  up
             tunnel-te2            200.0.0.3          up  up
             tunnel-te50           200.0.0.3          up  up
           *tunnel-te60         200.0.0.3           up  up
              *tunnel-te70         200.0.0.3          up  up
              *tunnel-te80         200.0.0.3          up  up
```

Related Topics

- Configuring Fast Reroute , on page 8
- Configuring Next Hop Backup Tunnel, on page 10
- MPLS-TE Features - Details, on page 35

Configuring Next Hop Backup Tunnel

The backup tunnels that bypass only a single link of the LSP path are referred as Next Hop (NHOP) backup tunnels because they terminate at the LSP's next hop beyond the point of failure. They protect LSPs, if a link along their path fails, by rerouting the LSP traffic to the next hop, thus bypassing the failed link.

Configuration Example

This example configures next hop backup tunnel on an interface and specifies the attribute-set template for the auto tunnels. In this example, unused backup tunnels are removed every 20 minutes using a timer and also the range of tunnel interface numbers are specified.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface HundredGigabitEthernet0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-mpls-te-if)# auto-tunnel backup nhop-only
RP/0/RP0/CPU0:router(config-mpls-te-if-auto-backup)# attribute-set ab
RP/0/RP0/CPU0:router(config-mpls-te)# auto-tunnel backup timers removal unused 20
RP/0/RP0/CPU0:router(config-mpls-te)# auto-tunnel backup tunnel-id min 6000 max 6500
RP/0/RP0/CPU0:router(config)# commit
```

Related Topics

- Configuring Auto-Tunnel Backup, on page 9
- Configuring Fast Reroute , on page 8
- MPLS-TE Features - Details, on page 35

Configuring SRLG Node Protection

Shared Risk Link Groups (SRLG) in MPLS traffic engineering refer to situations in which links in a network share common resources. These links have a shared risk, and that is when one link fails, other links in the group might fail too.

OSPF and IS-IS flood the SRLG value information (including other TE link attributes such as bandwidth availability and affinity) using a sub-type length value (sub-TLV), so that all routers in the network have the SRLG information for each link.

MPLS-TE SRLG feature enhances backup tunnel path selection by avoiding using links that are in the same SRLG as the interfaces it is protecting while creating backup tunnels.

Configuration Example

This example creates a backup tunnel and excludes the protected node IP address from the explicit path.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# interface HundredGigabitEthernet0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-mpls-te-if)# backup-path tunnl-te 2
RP/0/RP0/CPU0:router(config-mpls-te-if)# exit
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered Loopback0
RP/0/RP0/CPU0:router(config-if)# path-option 1 explicit name backup-srlg
RP/0/RP0/CPU0:router(config-if)# destination 192.168.92.125
RP/0/RP0/CPU0:router(config-if)# exit
RP/0/RP0/CPU0:router(config)# explicit-path name backup-srlg-nodep
RP/0/RP0/CPU0:router(config-if)# index 1 exclude-address 192.168.91.1
RP/0/RP0/CPU0:router(config-if)# index 2 exclude-srlg 192.168.92.2
RP/0/RP0/CPU0:router(config)# commit
```

Related Topics

- Configuring Fast Reroute , on page 8
- MPLS-TE Features - Details, on page 35

Configuring Pre-Standard DS-TE

Regular traffic engineering does not provide bandwidth guarantees to different traffic classes. A single bandwidth constraint is used in regular TE that is shared by all traffic. MPLS DS-TE enables you to configure multiple bandwidth constraints on an MPLS-enabled interface. These bandwidth constraints can be treated differently based on the requirement for the traffic class using that constraint. Cisco IOS XR software supports two DS-TE modes: Pre-standard and IETF. Pre-standard DS-TE uses the Cisco proprietary mechanisms for RSVP signaling and IGP advertisements. This DS-TE mode does not interoperate with third-party vendor equipment. Pre-standard DS-TE is enabled only after configuring the sub-pool bandwidth values on MPLS-enabled interfaces.

Pre-standard Diff-Serve TE mode supports a single bandwidth constraint model a Russian Doll Model (RDM) with two bandwidth pools: global-pool and sub-pool.

Before You Begin

The following prerequisites are required to configure a Pre-standard DS-TE tunnel.

- You must have a router ID for the neighboring router.
- Stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

Configuration Example

This example configures a pre-standard DS-TE tunnel.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# rsvp interface HundredGigabitEthernet 0/9/0/0
RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth 100 150 sub-pool 50
RP/0/RP0/CPU0:router(config-rsvp-if)# exit
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# signalled bandwidth sub-pool 10
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Use the **show mpls traffic-eng topology** command to verify the pre-standard DS-TE tunnel configuration.

Related Topics

- Configuring an IETF DS-TE Tunnel Using RDM, on page 12
- Configuring an IETF DS-TE Tunnel Using MAM, on page 13
- MPLS-TE Features - Details, on page 35

Configuring an IETF DS-TE Tunnel Using RDM

IETF DS-TE mode uses IETF-defined extensions for RSVP and IGP. This mode interoperate with third-party vendor equipment.

IETF mode supports multiple bandwidth constraint models, including Russian Doll Model (RDM) and Maximum Allocation Model (MAM), both with two bandwidth pools. In an IETF DS-TE network, identical bandwidth constraint models must be configured on all nodes.

Before you Begin

The following prerequisites are required to create a IETF mode DS-TE tunnel using RDM:

- You must have a router ID for the neighboring router.
- Stable router ID is required at either end of the link to ensure that the link is successful. If you do not assign a router ID to the routers, the system defaults to the global router ID. Default router IDs are subject to change, which can result in an unstable link.

Configuration Example

This example configures an IETF DS-TE tunnel using RDM.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# rsvp interface HundredGigabitEthernet 0/9/0/0
RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth rdm 100 150
RP/0/RP0/CPU0:router(config-rsvp-if)# exit
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# ds-te mode ietf
RP/0/RP0/CPU0:router(config-mpls-te)# exit
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# signalled bandwidth sub-pool 10 class-type 1
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Use the **show mpls traffic-eng topology** command to verify the IETF DS-TE tunnel using RDM configuration.

Related Topics

- Configuring Pre-Standard DS-TE, on page 11
- Configuring an IETF DS-TE Tunnel Using MAM, on page 13
- MPLS-TE Features - Details, on page 35

Configuring an IETF DS-TE Tunnel Using MAM

IETF DS-TE mode uses IETF-defined extensions for RSVP and IGP. This mode interoperates with third-party vendor equipment. IETF mode supports multiple bandwidth constraint models, including Russian Doll Model (RDM) and Maximum Allocation Model (MAM), both with two bandwidth pools.

Configuration Example

This example configures an IETF DS-TE tunnel using MAM.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# rsvp interface HundredGigabitEthernet 0/9/0/0
RP/0/RP0/CPU0:router(config-rsvp-if)# bandwidth mam max-reservable-bw 1000 bc0 600 bc1 400
RP/0/RP0/CPU0:router(config-rsvp-if)# exit
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# ds-te mode ietf
RP/0/RP0/CPU0:router(config-mpls-te)# ds-te bc-model mam
RP/0/RP0/CPU0:router(config-mpls-te)# exit
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# signalled bandwidth sub-pool 10
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Use the **show mpls traffic-eng topology** command to verify the IETF DS-TE tunnel using MAM configuration.

Related Topics

- Configuring an IETF DS-TE Tunnel Using RDM, on page 12

• Configuring Pre-Standard DS-TE, on page 11
• MPLS-TE Features - Details, on page 35

Configuring Flexible Name-Based Tunnel Constraints

MPLS-TE Flexible Name-based Tunnel Constraints provides a simplified and more flexible means of configuring link attributes and path affinities to compute paths for the MPLS-TE tunnels.

In traditional TE, links are configured with attribute-flags that are flooded with TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

MPLS-TE Flexible Name-based Tunnel Constraints lets you assign, or map, up to 32 color names for affinity and attribute-flag attributes instead of 32-bit hexadecimal numbers. After mappings are defined, the attributes can be referred to by the corresponding color name.

Configuration Example

This example shows assigning a how to associate a tunnel with affinity constraints.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# affinity-map red 1
RP/0/RP0/CPU0:router(config-mpls-te)# interface HundredGigabitEthernet0/0/0/30/9/0/0
RP/0/RP0/CPU0:router(config-mpls-te-if)# attribute-names red
RP/0/RP0/CPU0:router(config)# interface tunnel-te 2
RP/0/RP0/CPU0:router(config-if)# affinity include red
RP/0/RP0/CPU0:router(config)# commit
```

Configuring Automatic Bandwidth

Automatic bandwidth allows you to dynamically adjust bandwidth reservation based on measured traffic. MPLS-TE automatic bandwidth monitors the traffic rate on a tunnel interface and resizes the bandwidth on the tunnel interface to align it closely with the traffic in the tunnel. MPLS-TE automatic bandwidth is configured on individual Label Switched Paths (LSPs) at every headend router.

The following table specifies the parameters that can be configured as part of automatic bandwidth configuration.

Table 1: Automatic Bandwidth Parameters

Bandwidth Parameters	Description
Application frequency	Configures how often the tunnel bandwidths changed for each tunnel. The default value is 24 hours.
Bandwidth limit	Configures the minimum and maximum automatic bandwidth to set on a tunnel.
Bandwidth collection frequency	Enables bandwidth collection without adjusting the automatic bandwidth. The default value is 5 minutes.
Overflow threshold	Configures tunnel overflow detection.

Bandwidth Parameters	Description
Adjustment threshold	Configures the tunnel-bandwidth change threshold to trigger an adjustment.

Configuration Example

This example enables automatic bandwidth on MPLS-TE tunnel interface and configure the following automatic bandwidth variables.

- Application frequency
- Bandwidth limit
- Adjustment threshold
- Overflow detection

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface tunnel-te 1
RP/0/RP0/CPU0:router(config-if)# auto-bw
RP/0/RP0/CPU0:router(config-if-tunte-autobw)# application 1000
RP/0/RP0/CPU0:router(config-if-tunte-autobw)# bw-limit min 30 max 1000
RP/0/RP0/CPU0:router(config-if-tunte-autobw)# adjustment-threshold 50 min 800
RP/0/RP0/CPU0:router(config-if-tunte-autobw)# overflow threshold 100 limit 1
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Verify the automatic bandwidth configuration using the **show mpls traffic-eng tunnels auto-bw brief** command.

```
RP/0/RP0/CPU0:router# show mpls traffic-eng tunnels auto-bw brief

  Tunnel    LSP  Last appl  Requested  Signalled    Highest    Application
  Name       ID   BW(kbps)   BW(kbps)   BW(kbps)   BW(kbps)      Time Left
  ------------- ------ ---------- ---------- ---------- ---------- --------------
  tunnel-te1      5          500         300        420          1h 10m
```

Related Topics

- MPLS-TE Features - Details, on page 35

Autoroute Announce

The segment routing tunnel can be advertised into an Interior Gateway Protocol (IGP) as a next hop by configuring the autoroute announce statement on the source router. The IGP then installs routes in the Routing Information Base (RIB) for shortest paths that involve the tunnel destination. Autoroute announcement of IPv4 prefixes can be carried through either OSPF or IS-IS. Autoroute announcement of IPv6 prefixes can be carried only through IS-IS. IPv6 forwarding over tunnel needs additional parameter in tunnel configuration.

Restrictions

The Autoroute Announce feature is supported with the following restrictions:

- A maximum on 128 tunnels can be announced.
- Tunnels in ECMP may not be path diverse.



Note Configuring Segment Routing and Autoroute Destination together is not supported. If autoroute functionality is required in an Segment Routing network, we recommend you to configure Autoroute Announce.

Configuration

To specify that the Interior Gateway Protocol (IGP) should use the tunnel (if the tunnel is up) in its enhanced shortest path first (SPF) calculation, use the **autoroute announce** command in interface configuration mode.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# interface tunnel-te 10
RP/0/RP0/CPU0:router(config-if)# autoroute announce [include-ipv6]
RP/0/RP0/CPU0:router(config-if)# commit
```

Verification

Verify the route using the following commands:

```
RP/0/RP0/CPU0:router# Show route
Mon Aug  8 00:31:29.406 UTC

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, (!) - FRR Backup path

Gateway of last resort is not set

i L1 9.1.0.0/24 [115/30] via 9.9.9.9, 2d19h, tunnel-te3195
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3196
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3197
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3198
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3199
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3200
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3201
                [115/30] via 9.9.9.9, 2d19h, tunnel-te3202
i L1 9.9.9.9/32 [115/30] via 9.9.9.9, 2d22h, tunnel-te3195
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3196
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3197
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3198
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3199
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3200
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3201
                [115/30] via 9.9.9.9, 2d22h, tunnel-te3202

RP/0/RP0/CPU0:router# show route ipv6
Tue Apr 26 04:02:07.968 UTC

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, (!) - FRR Backup path

Gateway of last resort is not set

L    ::ffff:127.0.0.0/104
      [0/0] via ::, 00:17:36
C    2002::/64 is directly connected,
      00:39:01, TenGigE0/0/0/2
L    2002::1/128 is directly connected,
      00:39:01, TenGigE0/0/0/2
i L1 2003::/64
      [115/20] via ::, 00:02:32, tunnel-te992
      [115/20] via ::, 00:02:32, tunnel-te993
      [115/20] via ::, 00:02:32, tunnel-te994
      [115/20] via ::, 00:02:32, tunnel-te995
      [115/20] via ::, 00:02:32, tunnel-te996
      [115/20] via ::, 00:02:32, tunnel-te997
      [115/20] via ::, 00:02:32, tunnel-te998
      [115/20] via ::, 00:02:32, tunnel-te999
C    2006::/64 is directly connected,
      00:39:01, TenGigE0/0/0/6
L    2006::1/128 is directly connected,
      00:39:01, TenGigE0/0/0/6
i L1 2007::/64
      [115/20] via ::, 00:02:32, tunnel-te992
      [115/20] via ::, 00:02:32, tunnel-te993
      [115/20] via ::, 00:02:32, tunnel-te994
      [115/20] via ::, 00:02:32, tunnel-te995
      [115/20] via ::, 00:02:32, tunnel-te996
      [115/20] via ::, 00:02:32, tunnel-te997
      [115/20] via ::, 00:02:32, tunnel-te998
      [115/20] via ::, 00:02:32, tunnel-te999
```

```
show cef ipv6 2007::5 hardware egress location 0/0/CPU0
Sun May  1 03:18:55.151 UTC 2007::/64, version 50, internal 0x1000001 0x0 (ptr 0x895e819c)
 [1], 0x0 (0x894d3678), 0x0 (0x0)  Updated May  1 03:13:49.066  Prefix Len 64, traffic index
 0, precedence n/a, priority 2
   via ::/128, tunnel-te1, 2 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8b5ea420 0x0]
    next hop ::/128
    local adjacency
   via ::/128, tunnel-te2, 2 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 1 NHID 0x0 [0x8b5ea5e0 0x0]
    next hop ::/128
    local adjacency


 LEAF - HAL pd context :
 sub-type : IPV6, ecd_marked:0, has_collapsed_ldi:0  collapse_bwalk_required:0, ecdv2_marked:0
 HW Walk:
LEAF:
    Handle: 0xaabbccdd type: 1 FEC handle: 0x8999fb18

    LWLDI:
        PI:0x894d3678 PD:0x894d36b8 rev:214 p-rev:212 ldi type:3
        FEC hdl: 0x8999fb18 fec index: 0x0(0) num paths:2, bkup: 0

        SHLDI:
            PI:0x893658e8 PD:0x89365968 rev:212 p-rev:0 flag:0x0
```

```
            FEC hdl: 0x8999fb18 fec index: 0x20000002(2) num paths: 2 bkup paths: 2
            Path:0 fec index: 0x20004000(16384) DSP:0x440 Dest fec index: 0x2000101c(4124)
            Path:1 fec index: 0x20004002(16386) DSP:0x440 Dest fec index: 0x2000101c(4124)
            Path:2 fec index: 0x20004001(16385) DSP:0x440 Dest fec index: 0x2000101c(4124)
            Path:3 fec index: 0x20004003(16387) DSP:0x440 Dest fec index: 0x2000101c(4124)

V6TE NH HAL PD context :
pdptr 0x8b5ea488, flags :1, index:0

              TE-NH:
              Flag: PHP HW Tun_O_Tun frr_active_chg te_protect_chg , PD(0x880d1640), Push:
 0, Swap: 0, Link: 0
            ifhandle:0x800001c llabel:24001 FEC hdl: 0x896d4eb8 fec index: 0x20001022(4130),
 SRTE fec hdl:(nil)

V6TE NH HAL PD context :
pdptr 0x8b5ea648, flags :1, index:0

              TE-NH:
              Flag: PHP HW , PD(0x880d12d0), Push: 0, Swap: 0, Link: 0x8b34d518
            ifhandle:0x8000024 llabel:24000 FEC hdl: 0x896d4238 fec index: 0x2000101c(4124),
 SRTE fec hdl:(nil)
```

Autoroute announce with IS-IS

Autoroute Announce (AA) with IS-IS is a feature that allows routers to install MPLS-TE tunnels to destinations based on the shortest path calculated by IS-IS, improving traffic engineering and path selection for anycast prefixes.

Table 2: Feature History Table

Feature Name	Release	Feature Description
MPLS-TE IPv6-only autoroute announce	Release 25.4.1	This feature allows you to disable IPv4 autoroute announce without turning off autoroute announce entirely. To achieve IPv6-only announcements over MPLS-TE tunnels, use the new **exclude-ipv4** option along with the **include-ipv6** option in the autoroute announce configuration. This feature introduces these changes: CLI: A new keyword, **exclude-ipv4**, has been added to the **autoroute announce** command.

Feature Name	Release	Feature Description
Autoroute announce with IS-IS for anycast prefixes	Release 7.5.4	We have enabled seamless migration from autoroute announce (AA) tunnels with OSPF to AA tunnels with IS-IS. This is possible because you can now use AA tunnels with IS-IS to calculate the underlying native IGP metric for anycast prefixes to select the shortest path and then select a tunnel on that shortest path. Previously, autoroute announce tunnels with IS-IS behaved differently as compared to OSPF. IS-IS used the autoroute announce metric whereas, OSPF uses the underlying native IGP cost. With this feature the router uses the same method of cost calculation in autoroute announce in both OSPF and IS-IS. This feature introduces these: • CLI: **anycast-prefer-igp-cost** • YANG Data model: New Xpaths for Cisco-IOS-XR-um-router-isis-cfg.yang and Cisco-IOS-XR-clns-isis-cfg.yang (see GitHub and Yang Data Navigator)

When you configure Autoroute Announce (AA) tunnels, IGP installs the tunnel to the destination in the Routing Information Base (RIB) for the shortest paths. However, the behavior of AA for anycast destination is different from IS-IS compared to OSPF.

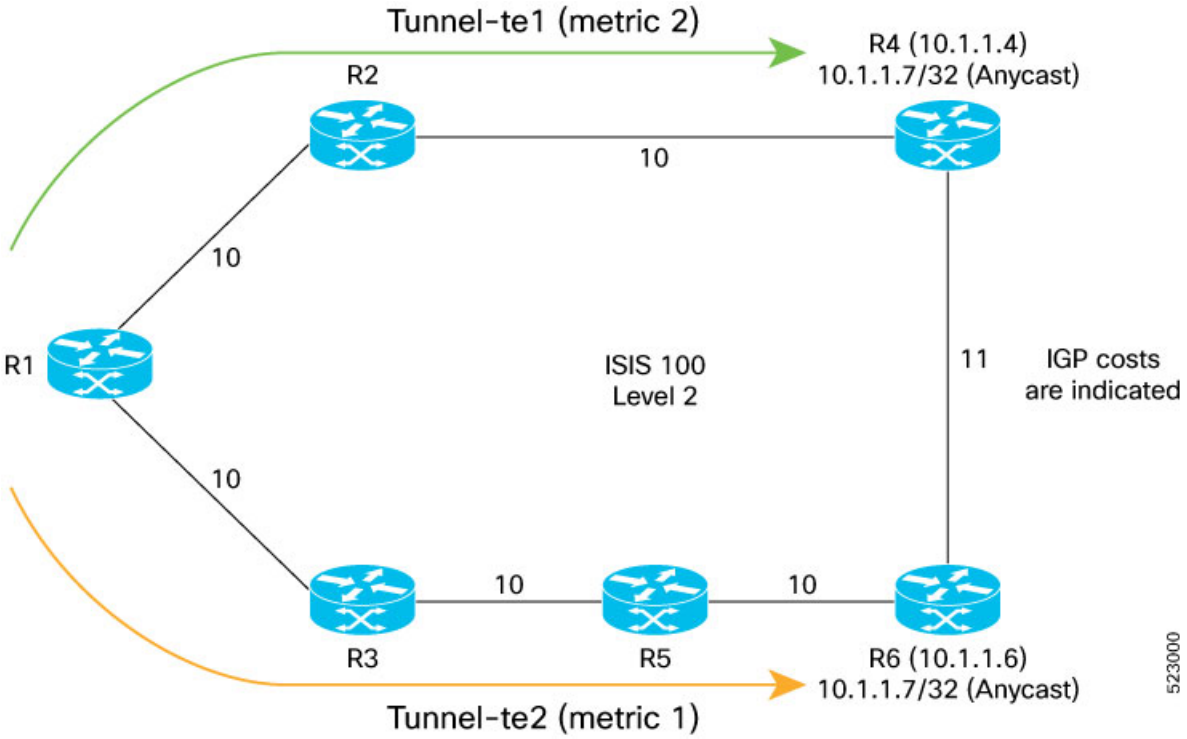
- IS-IS: AA tunnel with IS-IS considers the AA metric to calculate the prefix reachability for the shortest path.
- OSPF: AA with OSPF uses the lowest IGP cost to find the shortest path. And, then use the TE tunnel which is on that shortest path.

We now introduce the **anycast-prefer-igp-cost** command for AA tunnel with IS-IS anycast prefixes to allow the router to choose the shortest IGP path and select the tunnel on that shortest path, similar to OSPF behavior.

Starting from Cisco IOS XR Release 25.4.1, the autoroute announce feature restricts the usage of MPLS-TE tunnels by IGP to IPv6 prefixes. To enable the IPv6-only autoroute announce feature, configure the **autoroute announce exclude-ipv4** and **autoroute announce include-ipv6** commands.

Topology

Using this topology, let's see how the shortest path is chosen based on the IGP cost:



- In this topology, R1 is the headend and has MPLS RSVP-TE tunnels to R2 and R3.
- The tunnel destinations are loopback addresses of R4 and R6, which is 10.1.1.4 and 10.1.1.6.
- R1 uses AA to allow traffic to prefixes advertised by R2 and R3 to go over the MPLS-TE tunnel.
- Two AA tunnels are configured with different metrics and destinations. Tunnel-te1 has a metric value of 2 and tunnel-te2 has a metric value of 1.
- R1 can reach the anycast prefix (10.1.1.7/32) using two paths R1-R2-R4 and R1-R3-R5-R6.
- To reach the anycast prefix, the IGP cost is 20 for the R1-R2-R4 path and it is 30 for the R1-R3-R5-R6 path. Based on the IGP metric, the R1-R2-R4 path is the shortest path.
- With IS-IS, by default, the AA tunnel considers the tunnel metric to forward the traffic. In this topology, tunnel-te2 has a lower metric value when compared to tunnel-te1. As the AA metric is considered and the traffic is forwarded through te-2.
- When this feature is enabled, the IGP cost is first considered for the shortest path like AA with OSPF.

  In this topology, the R1-R2-R4 path is considered as the IGP cost is 20 that lower when compared to tunnel-te2, which is 30. Then the MPLS-TE tunnel on that path is considered, which is tunnel-te1.

Configure IPv6-only autoroute announce over MPLS-TE tunnels

Perform these steps to configure IPv6-only autoroute announce over MPLS-TE tunnels:

Procedure

Step 1 **configure**

Step 2 **interface tunnel-te** *tunnel-id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface tunnel-te 1
```

Configures an MPLS-TE tunnel interface.

Step 3 **autoroute announce include-ipv6**

Example:

```
RP/0/RP0/CPU0:router(config-if)# autoroute announce include-ipv6
```

Enables IPv6 autoroute announce.

Step 4 **exclude-ipv4**

Example:

```
RP/0/RP0/CPU0:router(config-if-tunte-aa)# exclude-ipv4
```

Disables IPv4 autoroute announce.

Step 5 **commit**

Step 6 (Optional) **show mpls traffic-eng autoroute**

Example:

```
RP/0/RP0/CPU0:router#show mpls traffic-eng autoroute

  Destination 192.168.0.2 has 1 tunnels in IS-IS ring level 1
    T1 (traffic share 0, nexthop 192.168.0.3,  metric 0)
    (IS-IS ring level-1, IPV6 Unicast)
    Signalled-Name: T1
```

Verifies that the tunnel announces IPv6-only autoroute information.

Configure IGP path selection for anycast prefixes

This procedure provides the steps to configure MPLS RSVP-TE and enable **anycast-prefer-igp-cost** with IS-IS.

Procedure

Step 1 Configure MPLS RSVP-TE.

Example:

```
/* Configure MPLS RSVP-TE */
Router(config)#interface tunnel-te1
```

```
Router(config-if)#ipv4 unnumbered Loopback0
Router(config-if)#autoroute announce
Router(config-if-tunte-aa)#metric 2
Router(config-if-tunte-aa)#exit
Router(config-if)#destination 10.1.1.4
Router(config-if)#path-option 1 dynamic
Router(config-if)#commit

Router(config)#interface tunnel-te2
Router(config-if)#ipv4 unnumbered Loopback0
Router(config-if)#autoroute announce
Router(config-if-tunte-aa)#metric 1
Router(config-if-tunte-aa)#exit
Router(config-if)#destination 10.1.1.6
Router(config-if)#path-option 1 dynamic
Router(config-if)#exit
Router(config)#commit
```

Step 2 Enable **anycast-prefer-igp-cost** with IS-IS.

Example:

```
Router#configure
Router(config)#router isis 100
Router(config-isis)#is-type level-2-only
Router(config-isis)#net 47.2377.50ea.ffff.988a.2d13.00
Router(config-isis)#address-family ipv4 unicast
Router(config-isis-af)#metric-style wide
Router(config-isis-af)#mpls traffic-eng level-2-only
Router(config-isis-af)#mpls traffic-eng router-id Loopback0
Router(config-isis-af)#mpls traffic-eng tunnel anycast-prefer-igp-cost
Router(config-isis-af)#maximum-paths 64
Router(config-isis-af)#commit
```

Step 3 Use the **show running config** to view the running configuration details.

Example:

```
Router# show running-config
interface tunnel-te1
 ipv4 unnumbered Loopback0
 autoroute announce
  metric 2
 !
 destination 10.1.1.4
 path-option 1 dynamic
!
interface tunnel-te2
 ipv4 unnumbered Loopback0
 autoroute announce
  metric 1
 !
 destination 10.1.1.6
 path-option 1 dynamic
!
router isis 100
 is-type level-2-only
 net 47.2377.50ea.ffff.988a.2d13.00
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  mpls traffic-eng tunnel anycast-prefer-igp-cost
  maximum-paths 64
```

```
 !
!
end
```

Step 4 Execute the **show route** command to verify the configuration.

- When you enable **anycast-prefer-igp-cost**, the traffic is forwarded through tunnel-te1:

```
Router#show route 10.1.1.7/32
Routing entry for 10.1.1.7/32
  Known via "isis 100", distance 115, metric 12, type level-2
  Installed Nov  3 09:48:39.520 for 00:00:05
  Routing Descriptor Blocks
    10.1.1.4, from 10.1.1.4, via tunnel-te1
      Route metric is 20
```

- When you disable **anycast-prefer-igp-cost**, the traffic is forwarded through tunnel-te2:

```
Router#show route 10.1.1.7/32
Routing entry for 10.1.1.7/32
  Known via "isis 100", distance 115, metric 11, type level-2
  Installed Nov  3 09:25:38.162 for 00:18:23
  Routing Descriptor Blocks
    10.1.1.6, from 10.1.1.6, via tunnel-te2
      Route metric is 11
```

Configure IPv6 routing over IPv4 MPLS-TE tunnels

Perform these steps to configure IPv6 routing over IPv4 MPLS-TE tunnels:

Procedure

Step 1 **configure**

Step 2 **interface tunnel-te** *tunnel-id*

Example:

```
RP/0/RP0/CPU0:router# interface tunnel-te 1
```

Configures an MPLS-TE tunnel interface.

Step 3 **ipv4 unnumbered** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-if)#ipv4 unnumbered Loopback 0
```

Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is the commonly-used interface type.

Step 4 **ipv6 enable**

Example:

```
RP/0/RP0/CPU0:router(config-if)#ipv6 enable
```

Enables IPv6 on interface.

Step 5 **signalled-bandwidth** *bandwidth*

Example:

```
RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 10
```

Sets the tunnel bandwidth requirement to be signalled in Kbps.

Step 6 **destination** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-if)#destination 192.168.0.1
```

Specifies the IPv4 destination address.

Step 7 Use one of these options:

- **autoroute announce include-ipv6**
- **forwarding-adjacency include-ipv6**

Example:

```
RP/0/RP0/CPU0:router(config-if)#autoroute announce include-ipv6
```

Or

```
RP/0/RP0/CPU0:router(config-if)#forwarding-adjacency include-ipv6
```

Announces the tunnel as an IPv6 autoroute, or as an IPv6 forwarding adjacency.

Step 8 **path-option** *preference-priority* **dynamic**

Example:

```
RP/0/RP0/CPU0:router(config-if)#path-option 1 dynamic
```

Sets the path option to dynamic and assigns the path ID.

Step 9 **commit**

Step 10 (Optional) **show mpls traffic-eng autoroute**

Example:

```
RP/0/RP0/CPU0:router#show mpls traffic-eng autoroute

  Destination 192.168.0.2 has 1 tunnels in IS-IS ring level 1
    tunnel-te1 (traffic share 0, nexthop 192.168.0.2)
               (IPv4 unicast)
               (IPv6 unicast)
```

Verifies that the tunnel announces IPv6 autoroute information.

Step 11 (Optional) **show mpls traffic-eng forwarding-adjacency**

Example:

```
RP/0/RP0/CPU0:router#show mpls traffic-eng forwarding-adjacency

 destination 192.168.0.1 has 1 tunnels

              tunnel-te10    (traffic share 0, next-hop 192.168.0.1)
                             (Adjacency Announced: yes, holdtime 0)
                             (IS-IS 100, IPv4 unicast)
                             (IS-IS 100, IPv6 unicast)
```

Verifies that the tunnel announces IPv6 forwarding adjacency information.

Configure IPv6 routing over IPv6 MPLS-TE tunnels

Perform these steps to configure IPv6 routing over IPv6 MPLS-TE tunnels:

Procedure

Step 1 **configure**

Step 2 **interface tunnel-te** *tunnel-id*

Example:

```
RP/0/RP0/CPU0:router# interface tunnel-te 1
```

Configures an MPLS-TE tunnel interface.

Step 3 **ipv4 unnumbered** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-if)#ipv4 unnumbered Loopback 0
```

Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is the commonly-used interface type.

Step 4 **ipv6 enable**

Example:

```
RP/0/RP0/CPU0:router(config-if)#ipv6 enable
```

Enables IPv6 on interface.

Step 5 **signalled-bandwidth** *bandwidth*

Example:

```
RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 10
```

Sets the tunnel bandwidth requirement to be signalled in Kbps.

Step 6 Use one of these options:

- **autoroute announce include-ipv6**
- **forwarding-adjacency include-ipv6**

Example:

```
RP/0/RP0/CPU0:router(config-if)#autoroute announce include-ipv6
```

Or

```
RP/0/RP0/CPU0:router(config-if)#forwarding-adjacency include-ipv6
```

Announces the tunnel as an IPv6 autoroute, or as an IPv6 forwarding adjacency.

Step 7 **destination ipv6** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-if)#destination ipv6 2000::1
```

Specifies the IPv6 destination address.

Step 8 **path-option** *preference-priority* **dynamic**

Example:

```
RP/0/RP0/CPU0:router(config-if)#path-option 1 dynamic
```

Sets the path option to dynamic and assigns the path ID.

Step 9 **commit**

Step 10 (Optional) **show mpls traffic-eng autoroute**

Example:

```
RP/0/RP0/CPU0:router#show mpls traffic-eng autoroute

  Destination 192.168.0.2 has 1 tunnels in IS-IS ring level 1
    tunnel-te1 (traffic share 0, nexthop 192.168.0.2)
               (IPv4 unicast)
               (IPv6 unicast)
```

Verifies that the tunnel announces IPv6 autoroute information.

Step 11 (Optional) **show mpls traffic-eng forwarding-adjacency**

Example:

```
RP/0/RP0/CPU0:router#show mpls traffic-eng forwarding-adjacency

 destination 192.168.0.1 has 1 tunnels

              tunnel-te10    (traffic share 0, next-hop 192.168.0.1)
                             (Adjacency Announced: yes, holdtime 0)
                             (IS-IS 100, IPv4 unicast)
                             (IS-IS 100, IPv6 unicast)
```

Verifies that the tunnel announces IPv6 forwarding adjacency information.

Configuring Auto-Tunnel Mesh

The MPLS-TE auto-tunnel mesh (auto-mesh) feature allows you to set up full mesh of TE Point-to-Point (P2P) tunnels automatically with a minimal set of MPLS traffic engineering configurations. You can configure one or more mesh-groups and each mesh-group requires a destination-list (IPv4 prefix-list) listing destinations, which are used as destinations for creating tunnels for that mesh-group.

You can configure MPLS-TE auto-mesh type attribute-sets (templates) and associate them to mesh-groups. Label Switching Routers (LSRs) can create tunnels using the tunnel properties defined in this attribute-set.

Auto-Tunnel mesh configuration minimizes the initial configuration of the network. You can configure tunnel properties template and mesh-groups or destination-lists on TE LSRs that further creates full mesh of TE tunnels between those LSRs. It eliminates the need to reconfigure each existing TE LSR in order to establish a full mesh of TE tunnels whenever a new TE LSR is added in the network.

Configuration Example

This example configures an auto-tunnel mesh group and specifies the attributes for the tunnels in the mesh-group.

```
RP/0/RP0/CPU0:router # configure
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# auto-tunnel mesh
RP/0/RP0/CPU0:router(config-mpls-te-auto-mesh)# tunnel-id min 1000 max 2000
RP/0/RP0/CPU0:router(config-mpls-te-auto-mesh)# group 10
RP/0/RP0/CPU0:router(config-mpls-te-auto-mesh-group)# attribute-set 10
RP/0/RP0/CPU0:router(config-mpls-te-auto-mesh-group)# destination-list dl-65
RP/0/RP0/CPU0:router(config-mpls-te)# attribute-set auto-mesh 10
RP/0/RP0/CPU0:router(config-mpls-te-attribute-set)# autoroute announce
RP/0/RP0/CPU0:router(config-mpls-te-attribute-set)# auto-bw collect-bw-only
RP/0/RP0/CPU0:router(config)# commit
```

Verification

Verify the auto-tunnel mesh configuration using the **show mpls traffic-eng auto-tunnel mesh** command.

```
RP/0/RP0/CPU0:router# show mpls traffic-eng auto-tunnel mesh

Auto-tunnel Mesh Global Configuration:
  Unused removal timeout: 1h 0m 0s
  Configured tunnel number range: 1000-2000

Auto-tunnel Mesh Groups Summary:
  Mesh Groups count: 1
  Mesh Groups Destinations count: 3
  Mesh Groups Tunnels count:
    3 created, 3 up, 0 down, 0 FRR enabled

Mesh Group: 10 (3 Destinations)
  Status: Enabled
  Attribute-set: 10
  Destination-list: dl-65 (Not a prefix-list)
  Recreate timer: Not running
       Destination    Tunnel ID    State  Unused timer
  ----------------  -----------  -------  ------------
       192.168.0.2         1000     up   Not running
       192.168.0.3         1001     up   Not running
       192.168.0.4         1002     up   Not running
  Displayed 3 tunnels, 3 up, 0 down, 0 FRR enabled

Auto-mesh Cumulative Counters:
  Last cleared: Wed Oct  3 12:56:37 2015 (02:39:07 ago)
                       Total
  Created:                 3
  Connected:               0
  Removed (unused):        0
  Removed (in use):        0
  Range exceeded:          0
```

Configuring an MPLS Traffic Engineering Interarea Tunneling

The MPLS TE Interarea Tunneling feature allows you to establish MPLS TE tunnels that span multiple Interior Gateway Protocol (IGP) areas and levels. This feature removes the restriction that required the tunnel headend and tailend routers both to be in the same area. The IGP can be either Intermediate System-to-Intermediate

System (IS-IS) or Open Shortest Path First (OSPF).To configure an inter-area tunnel, you specify on the headend router a loosely routed explicit path for the tunnel label switched path (LSP) that identifies each area border router (ABR) the LSP should traverse using the next-address loose command. The headend router and the ABRs along the specified explicit path expand the loose hops, each computing the path segment to the next ABR or tunnel destination.

Configuration Example

This example configures an IPv4 explicit path with ABR configured as loose address on the headend router.

```
Router# configure
Router(config)# explicit-path name interarea1
Router(config-expl-path)# index1 next-address loose ipv4 unicast 172.16.255.129
Router(config-expl-path)# index 2 next-address loose ipv4 unicast 172.16.255.131
Router(config)# interface tunnel-te1
Router(config-if)# ipv4 unnumbered Loopback0
Router(config-if)# destination 172.16.255.2
Router(config-if)# path-option 10 explicit name interarea1
Router(config)# commit
```

Configure Policy-Based Tunnel Selection

Configuring PBTS is a process of directing incoming traffic into specific TE tunnels based on a classification criteria (DSCP). The traffic forwarding decisions are made based on the categorized traffic classes and the destination network addresses. The following section lists the steps to configure PBTS on a MPLS-TE Tunnel network:

1. Define a class-map based on a classification criteria.
2. Define a policy-map by creating rules for the classified traffic.
3. Associate a forward-class to each type of ingress traffic.
4. Enable PBTS on the ingress interface, by applying this service-policy.
5. Create one or more egress MPLS-TE Tunnels (to carry packets based on priority) to the destination.
6. Associate the egress MPLS-TE Tunnel to a forward-class.

For more information on PBTS, see Policy-Based Tunnel Selection , on page 38 in the *Implementing MPLS Traffic Engineering* chapter.

Configuration Example

The following section illustrates PBTS implementation:

```
RP/0/RP0/CPU0:router#configure
/* Class-map; classification using DSCP */
RP/0/RP0/CPU0:router(config)# class-map match-any AF41-Class
RP/0/RP0/CPU0:router(config-cmap)# match dscp AF41
RP/0/RP0/CPU0:router(config-cmap)# exit

/* Policy-map */
RP/0/RP0/CPU0:router(config)# policy-map INGRESS-POLICY
RP/0/RP0/CPU0:router(config-pmap)# class AF41-Class
/* Associating forward class */
RP/0/RP0/CPU0:router(config-pmap-c)# set forward-class 1
RP/0/RP0/CPU0:router(config-pmap-c)# exit
RP/0/RP0/CPU0:router(config-pmap)# exit

RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/9/0/0
/* Applying service-policy to ingress interface */
```

```
RP/0/RP0/CPU0:router(config-if)# service-policy input INGRESS-POLICY
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
RP/0/RP0/CPU0:router(config-if)# exit

/* Creating TE-tunnels to carry traffic based on priority */
RP/0/RP0/CPU0:router(config)# interface tunnel-te61
RP/0/RP0/CPU0:router(config-if)# ipv4 unnumbered Loopback0
RP/0/RP0/CPU0:router(config-if)# signalled-bandwidth 1000
RP/0/RP0/CPU0:router(config-if)# autoroute announce
RP/0/RP0/CPU0:router(config-if)# destination 10.20.20.1
RP/0/RP0/CPU0:router(config-if)# record route

/* Associating egress TE tunnels to forward class */
RP/0/RP0/CPU0:router(config-if)# forward-class 1
RP/0/RP0/CPU0:router(config-if)# path-option 1 explicit identifier 61
RP/0/RP0/CPU0:router(config-if)# exit
```

Verification

Use **show mpls forwarding tunnels** command to verify the PBTS configuration:

```
RP/0/RP0/CPU0:ios# show mpls forwarding tunnels 10 detail
Tue May 16 01:18:19.681 UTC

Tunnel        Outgoing    Outgoing     Next Hop        Bytes
Name          Label       Interface                    Switched
------------- ----------- ------------ --------------- ------------
tt10              Exp-Null-v4 Te0/0/0/16   20.20.17.21     0
     Updated: May 11 19:31:54.716
     Version: 483, Priority: 2
     Label Stack (Top -> Bottom): { 0 }
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 14/18, MTU: 1500
     Packets Switched: 0

  Interface:
    Name: tunnel-te10 (ifhandle 0x0800005c)
    Local Label: 64016, Forwarding Class: 1, Weight: 0

    Packets/Bytes Switched: 0/0
```

Configuring Auto-bandwidth Bundle TE++

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Automatic Bandwidth Bundle TE++ for Numbered Tunnels	Release 7.10.1	We have optimized network performance and enabled efficient utilization of resources for numbered tunnels based on real-time traffic by automatically adding or removing tunnels between two endpoints. This is made possible because this release introduces support for auto-bandwidth TE++ for numbered tunnels, expanding upon the previous support for only named tunnels, letting you define explicit paths and allocate the bandwidth to each tunnel. The feature introduces these changes: • **CLI:** The **auto-capacity** keyword is added to the **interface tunnel-te** command. • **YANG Data Model:** New XPaths for `Cisco-IOS-XR-mpls-te-cfg.yang` (see GitHub, YANG Data Models Navigator)

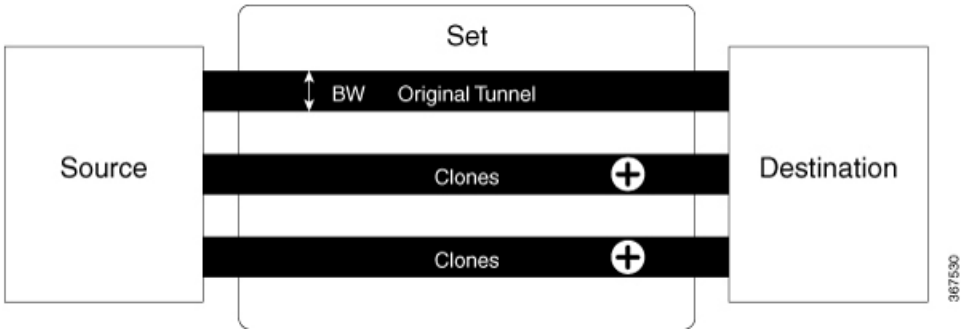
MPLS-TE tunnels are used to set up labeled connectivity and to provide dynamic bandwidth capacity between endpoints. The auto-bandwidth feature addresses the dynamic bandwidth capacity demands by dynamically resizing the MPLS-TE tunnels based on the measured traffic loads. However, many customers require multiple auto-bandwidth tunnels between endpoints for load balancing and redundancy. When the aggregate bandwidth demand increases between two endpoints, you can either configure auto-bandwidth feature to resize the tunnels or create new tunnels and load balance the overall demand over all the tunnels between two endpoints. Similarly, when the aggregate bandwidth demand decreases between two endpoints you can either configure the auto-bandwidth feature to decrease the sizes of the tunnel or delete the new tunnels and load balance the traffic over the remaining tunnels between the endpoints. The autobandwidth bundle TE++ feature is an extension of the auto-bandwidth feature and allows you to automatically increase or decrease the number of MPLS-TE tunnels to a destination based on real time traffic needs.

Tunnels that are automatically created as a response to the increasing bandwidth demands are called clones. The cloned tunnels inherit properties of the main configured tunnel. However, user configured load interval

cannot be inherited. The original tunnel and its clones are collectively called a set. You can specify an upper limit and lower limit on the number of clones that can be created for the original tunnel.

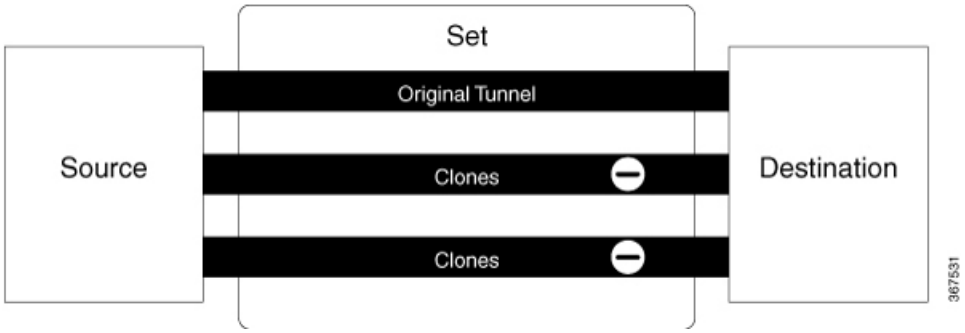
Splitting is the process of cloning a new tunnel when there is a demand for bandwidth increase. When the size of any of the tunnels in the set crosses a configured split bandwidth, then splitting is initiated and clone tunnels are created.

The following figure explains creating clone tunnels when the split bandwidth is exceeded.



Merging is the process of removing a clone tunnel when the bandwidth demand decreases. If the bandwidth goes below the configured merge bandwidth in any one of the tunnels in the set, clone tunnels are removed.

The following figure explains removing clone tunnels to merge with the original tunnel when the bandwidth falls below the merge bandwidth.



There are multiple ways to equally load-share the aggregate bandwidth demand among the tunnels in the set. This means that an algorithm is needed to choose the pair which satisfies the aggregate bandwidth requirements. You can configure a nominal bandwidth to guide the algorithm to determine the average bandwidths of the tunnels. If nominal bandwidth is not configured, TE uses the average of split and merge bandwidth as nominal bandwidth.

Restrictions and Usage Guidelines

The following usage guidelines apply for the auto-bandwidth bundle TE++ feature.

- This feature is only supported for the named tunnels and not supported on tunnel-te interfaces.
- The range for the lower limit on the number of clones is 0 to 63 and the default value for the lower limit on the number of clones is 0.
- The range for the upper limit on the number of clones is 1 to 63 and the default value for the upper limit on the number of clones is 63.

Configuration Example

This example shows how to configure the autobandwidth bundle TE++ feature for a named MPLS-TE traffic tunnel. You should configure the following values for this feature to work:

- min-clones: Specifies the minimum number of clone tunnels that the original tunnel can create.
- max-clones: Specifies the maximum number of clone tunnels that the original tunnel can create.
- nominal-bandwidth: Specifies the average bandwidth for computing the number of tunnels to satisfy the overall demand.
- split-bandwidth: Specifies the bandwidth value for splitting the original tunnel. If the tunnel bandwidth exceeds the configured split bandwidth, clone tunnels are created.
- merge-bandwidth: Specifies the bandwidth for merging clones with the original tunnel. If the bandwidth goes below the configured merge bandwidth, clone tunnels are removed.

In this example, the lower limit on the number of clones is configured as two and the upper limit on the number of clones is configured as four. The bandwidth size for splitting and merging is configured as 200 and 100 kbps.

```
RP/0/RP0/CPU0:router(config)# mpls traffic-eng
RP/0/RP0/CPU0:router(config-mpls-te)# named-tunnels
RP/0/RP0/CPU0:router(config-te-named-tunnels)# tunnel-te xyz
RP/0/RP0/CPU0:router(config-te-tun-name)# auto-bw
RP/0/RP0/CPU0:router(config-mpls-te-tun-autobw)# auto-capacity
RP/0/RP0/CPU0:router(config-te-tun-autocapacity)# min-clones 2
RP/0/RP0/CPU0:router(config-te-tun-autocapacity)# max-clones 4
RP/0/RP0/CPU0:router(config-te-tun-autocapacity)# nominal-bandwidth 150
RP/0/RP0/CPU0:router(config-te-tun-autocapacity)# split-bandwidth 200
RP/0/RP0/CPU0:router(config-te-tun-autocapacity)# merge-bandwidth 100
```

Regular expressions in MPLS-TE tunnel names

In large scale MPLS-TE tunnel deployments, regular expressions (regex) can provide a scalable way to filter tunnels for visibility, run actions on group of tunnels, and avoid issuing commands one tunnel at a time.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Regular expressions in MPLS-TE tunnel names	Release 26.2.1	This feature enhances operational scalability and usability by allowing regex-based tunnel name filtering for MPLS-TE show and execution commands, without changing configuration models, architecture, or security posture. This feature supports regular expression in all MPLS-TE show and execution commands using tunnel names.

Starting from Cisco IOS XR Release 26.2.1, you can use regular expressions in execution and show commands with MPLS-TE tunnel names to filter multiple tunnels at a time.

Regular expressions enable operators to quickly filter show command output for MPLS-TE tunnel names, allowing them to

- view all tunnels for a specific region, customer, or service
- reduce noise in large outputs, and
- troubleshoot issues more quickly during outages or convergence events

By using regular expressions, operators can identify all related MPLS-TE tunnels at once, streamlining operational tasks across large-scale deployments.

Benefits of using regular expressions in MPLS-TE tunnel names

- Manage all related tunnels at once in large scale deployments.
- Apply actions to multiple tunnels in a single command, such as reoptimizing paths, rerouting tunnels, and clearing auto-bandwidth state.
- Leverage existing naming conventions without changing tunnel configuration.
- Allow fine-grained selection preventing incorrect matches in large MPLS-TE networks.

Limitations of using regex in MPLS-TE tunnels

The gNMI, Netconf, and programmability interfaces do not support regular expressions.

Usage guidelines of regex in MPLS-TE tunnel names

Follow these guidelines to use regex with MPLS-TE tunnel names:

- Tunnel names support the dot character ('.'). However, using `.` as a regex in tunnel names does not match any single character. To filter tunnel names in this way, use regex patterns such as `b[0-9a-zA-Z_-]r` or `b(.)r`.

  `.*` and `.+` are exceptions that are considered as regular expressions matching any character N times.

- When your regex includes '(', ')', or '|', enclose the pattern in double quotes. Without double quotes, it will be reported as an unsupported character.

```
Router# mpls traffic-eng reoptimize name "(FROM-nyc.*)"
Router# mpls traffic-eng reoptimize name "(FROM-nyc|To-LA)"
```

- Regex patterns containing backslash (\) do not work because the IOS XR parser removes all backslash characters from the string before processing.
  - Use `[0-9]` instead of `\d`
  - Use `[A-Za-z0-9_]` instead of `\w`
  - Using a dot (.) matches literal `'.'`, and there is no need to use `\.`

    Example using `\w`:

```
Router# show mpls traffic-eng tunnels name fo.\w brief
TUNNEL NAME         DESTINATION      STATUS  STATE
       fo.w         192.168.0.3          up     up
```

Example using .(dot):

```
Router# show mpls traffic-eng tunnels name fo.o brief

TUNNEL NAME         DESTINATION      STATUS  STATE
       fo.o         192.168.0.3          up     up

Router# show mpls traffic-eng tunnels name fo[.f]o brief

TUNNEL NAME         DESTINATION      STATUS  STATE
       fo.o         192.168.0.3          up     up
       fofo         192.168.0.3          up     up
```

- For backward compatibility, all patterns are automatically wrapped as `^pattern$`.
- To match substrings, explicitly include wildcards such as `.*pattern.*`.

  Example:

  ```
  /* Matches only the tunnel name "foo". */
  Router# show mpls traffic-eng tunnels name foo

  /* Matches all tunnels containing "foo" */
  Router# show mpls traffic-eng tunnels name .*foo.*
  ```

- The question mark ('?') is interpreted by the CLI for command help or auto-completion. Hence, avoid using `?` in regex.

Table 5: Regex summary

Typical regex	Parser behavior	Usage
`'.'`	Literal only	Use `'.'` or character class such as `[a-zA-Z0-9_-.]`
`'(', ')', '|'`	Not allowed	Not allowed unless inside double quotes
`\`	Stripped	Use explicit character classes
`?`	CLI help character	Do not use
`^$`	Auto-applied	Explicit use has no impact
Substring match	Disabled by default	Use `.*pattern.*`

By following these guidelines, you ensure your regex patterns for MPLS-TE tunnel names are parsed and executed as intended in show and execution commands.

Regular expressions in MPLS-TE tunnel name filtering commands

You can filter and manage MPLS-TE tunnels efficiently by specifying name patterns with regular expressions in CLI commands. Regular expressions allow you to match tunnels based on their naming patterns for more effective tunnel management.

Supported show commands and execution commands

Execution commands	Show commands
mpls traffic-eng reoptimize name *<tunn-name with regex>*	**show mpls traffic-eng tunnels name** *<tunn-name with regex>* **[affinity | auto-bw | brief | detail | fsm | name-lookup | signame | source | standby | tabular]**
mpls traffic-eng tunnels resetup name *<tunn-name with regex>*	**show mpls traffic-eng event-history tunnels name** *<tunn-name with regex>*
mpls traffic-eng tunnels reroute name *<tunn-name with regex>*	**show mpls traffic-eng autoroute name** *<tunn-name with regex>* **[ip | standby]**
mpls traffic-eng tunnels reroute name *<tunn-name with regex>*	**show mpls traffic-eng autoroute -destination name** *<tunn-name with regex>* **[ip | standby]**
clear mpls traffic-eng auto-bw name *<tunn-name with regex>*	**show mpls traffic-eng tunnels protection [frr] name** *<tunn-name with regex>* **[name-lookup | standby]**
	show mpls traffic-eng tunnels name *<tunn-name with regex>* **role [all | head | middle | tail]**

Some examples include:

- Execution commands:

```
Router# mpls traffic-eng reoptimize name FROM-nyc.*
Router# mpls traffic-eng tunnels reroute name .*TO-LA
```

- Show commands:

```
Router# show mpls traffic-eng tunnels name FROM-nyc.*
Router# show mpls traffic-eng tunnels name .*TO-LA
Router# show mpls traffic-eng tunnels name .*CUSTOMER-A.*
```

These examples demonstrate how to use regular expressions, such as '.*', to match tunnels that begin with , end with, or contain specific name patterns.

MPLS-TE Features - Details

MPLS TE Fast Reroute Link and Node Protection

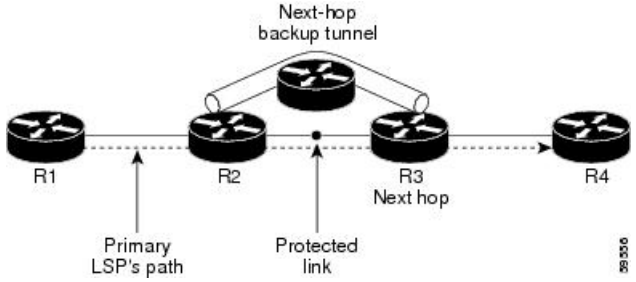
Fast Reroute (FRR) is a mechanism for protecting MPLS TE LSPs from link and node failures by locally repairing the LSPs at the point of failure, allowing data to continue to flow on them while their headend routers try to establish new end-to-end LSPs to replace them. FRR locally repairs the protected LSPs by rerouting them over backup tunnels that bypass failed links or node.

Note If FRR is greater than 50ms, it might lead to a loss of traffic.

Backup tunnels that bypass only a single link of the LSP's path provide link protection. They protect LSPs if a link along their path fails by rerouting the LSP's traffic to the next hop (bypassing the failed link). These tunnels are referred to as next-hop (NHOP) backup tunnels because they terminate at the LSP's next hop beyond the point of failure.

The following figure illustrates link protection.

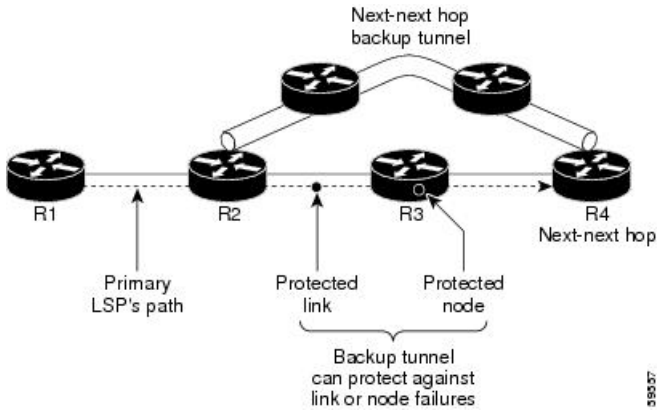
Figure 2: Link Protection



FRR provides node protection for LSPs. Backup tunnels that bypass next-hop nodes along LSP paths are called next-next-hop (NNHOP) backup tunnels because they terminate at the node following the next-hop node of the LSP paths, bypassing the next-hop node. They protect LSPs if a node along their path fails by enabling the node upstream of the failure to reroute the LSPs and their traffic around the failed node to the next-next hop. NNHOP backup tunnels also provide protection from link failures, because they bypass the failed link and the node.

The following figure illustrates node protection.

Figure 3: Node Protection



MPLS-TE Forwarding Adjacency

MPLS TE forwarding adjacency allows you to handle a TE label-switched path (LSP) tunnel as a link in an Interior Gateway Protocol (IGP) network that is based on the Shortest Path First (SPF) algorithm. Both Intermediate System-to-Intermediate System (IS-IS) and Open Shortest Path First (OSPF) are supported as the IGP. A forwarding adjacency can be created between routers regardless of their location in the network. The routers can be located multiple hops from each other.

As a result, a TE tunnel is advertised as a link in an IGP network with the tunnel's cost associated with it. Routers outside of the TE domain see the TE tunnel and use it to compute the shortest path for routing traffic throughout the network. TE tunnel interfaces are advertised in the IGP network just like any other links.

Routers can then use these advertisements in their IGPs to compute the SPF even if they are not the headend of any TE tunnels.

Automatic Bandwidth

Automatic bandwidth allows you to dynamically adjust bandwidth reservation based on measured traffic. MPLS-TE automatic bandwidth is configured on individual Label Switched Paths (LSPs) at every headend router. MPLS-TE automatic bandwidth monitors the traffic rate on a tunnel interface and resizes the bandwidth on the tunnel interface to align it closely with the traffic in the tunnel.

MPLS-TE automatic bandwidth can perform these functions:

- Monitors periodic polling of the tunnel output rate
- Resizes the tunnel bandwidth by adjusting the highest rate observed during a given period.

For every traffic-engineered tunnel that is configured for an automatic bandwidth, the average output rate is sampled, based on various configurable parameters. Then, the tunnel bandwidth is readjusted automatically based on either the largest average output rate that was noticed during a certain interval, or a configured maximum bandwidth value.

While re-optimizing the LSP with the new bandwidth, a new path request is generated. If the new bandwidth is not available, the last good LSP remains used. This way, the network experiences no traffic interruptions. If minimum or maximum bandwidth values are configured for a tunnel, the bandwidth, which the automatic bandwidth signals, stays within these values.

The output rate on a tunnel is collected at regular intervals that are configured by using the **application** command in MPLS-TE auto bandwidth interface configuration mode. When the application period timer expires, and when the difference between the measured and the current bandwidth exceeds the adjustment threshold, the tunnel is re-optimized. Then, the bandwidth samples are cleared to record the new largest output rate at the next interval. If a tunnel is shut down, and is later brought again, the adjusted bandwidth is lost, and the tunnel is brought back with the initially configured bandwidth. When the tunnel is brought back, the application period is reset.

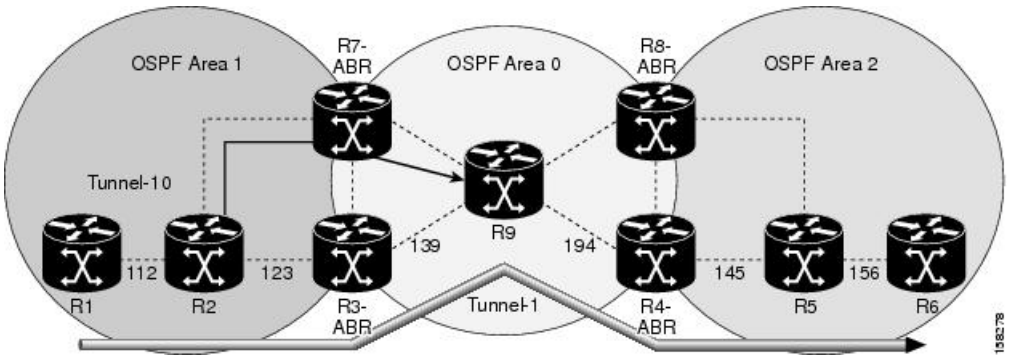
MPLS Traffic Engineering Interarea Tunneling

The MPLS-TE interarea tunneling feature allows you to establish TE tunnels spanning multiple Interior Gateway Protocol (IGP) areas and levels, thus eliminating the requirement that headend and tailend routers reside in a single area.

Interarea support allows the configuration of a TE LSP that spans multiple areas, where its headend and tailend label switched routers (LSRs) reside in different IGP areas. Customers running multiple IGP area backbones (primarily for scalability reasons) requires Multiarea and Interarea TE . This lets you limit the amount of flooded information, reduces the SPF duration, and lessens the impact of a link or node failure within an area, particularly with large WAN backbones split in multiple areas.

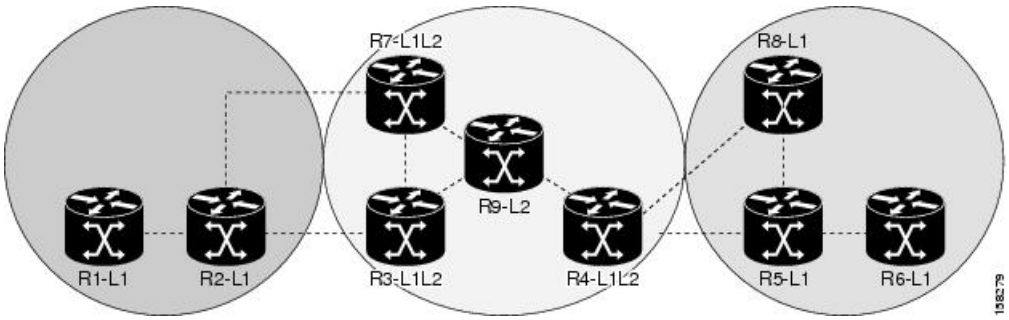
The following figure shows a typical interarea TE network using OSPF.

Figure 4: Interarea (OSPF) TE Network Diagram



The following figure shows a typical interlevel (IS-IS) TE Network.

Figure 5: Interlevel (IS-IS) TE Network Diagram



As shown in the Figure 5: Interlevel (IS-IS) TE Network Diagram, on page 38, R2, R3, R7, and R4 maintain two databases for routing and TE information. For example, R3 has TE topology information related to R2, flooded through Level-1 IS-IS LSPs plus the TE topology information related to R4, R9, and R7, flooded as Level 2 IS-IS Link State PDUs (LSPs) (plus, its own IS-IS LSP).

Loose hop optimization allows the re-optimization of tunnels spanning multiple areas and solves the problem which occurs when an MPLS-TE LSP traverses hops that are not in the LSP's headend's OSPF area and IS-IS level. Interarea MPLS-TE allows you to configure an interarea traffic engineering (TE) label switched path (LSP) by specifying a loose source route of ABRs along the path. Then it is the responsibility of the ABR (having a complete view of both areas) to find a path obeying the TE LSP constraints within the next area to reach the next hop ABR (as specified on the headend router). The same operation is performed by the last ABR connected to the tailend area to reach the tailend LSR.

You must be aware of these considerations when using loose hop optimization:

- You must specify the router ID of the ABR node (as opposed to a link address on the ABR).
- When multiarea is deployed in a network that contains subareas, you must enable MPLS-TE in the subarea for TE to find a path when loose hop is specified.
- You must specify the reachable explicit path for the interarea tunnel.

Policy-Based Tunnel Selection

Policy-Based Tunnel Selection (PBTS) is a mechanism that lets you direct traffic into specific TE tunnels based on different classification criteria. PBTS will benefit Internet service providers (ISPs) that carry voice

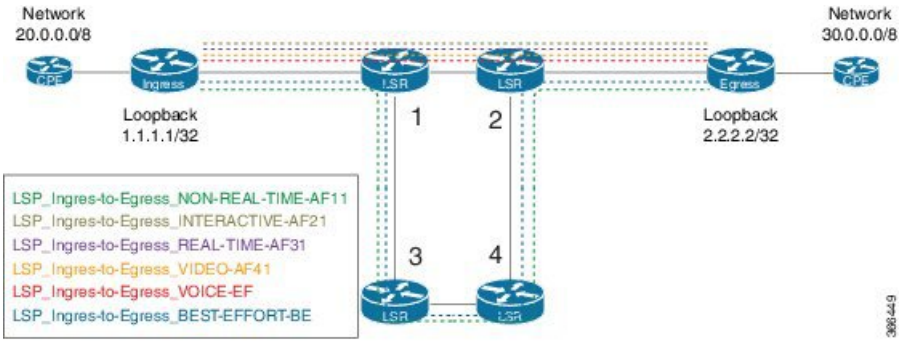
and data traffic through their MPLS and MPLS/VPN networks and would have to route this traffic to provide optimized voice service.

PBTS works by selecting tunnels based on the classification criteria of the incoming packets, which are based on the IP precedence or differentiated services code point (DSCP), or the Type of Service (ToS) fields in the packets. The traffic forwarding decisions are made based on the traffic classes AND the destination network addresses instead of only considering the destination network.

Default-class configured for paths is always zero (0). If there is no TE for a given forward-class, then the default-class (0) will be tried. If there is no default-class, then the packet is tried against the lowest configured forward-class tunnels. PBTS supports up to seven (exp 1 - 7) EXP values associated with a single TE-tunnel.

The following figure illustrates PBTS Network Topology:

Figure 6: Policy-Based Tunnel Selection Implementation



- Tunnels are created between Ingress and Egress nodes through LSR 1-2 and LSR 1-3-4-2 paths.
- High priority traffic takes the path: Ingress->LSR1->LSR2->Egress.
- Low priority traffic takes the path: Ingress->LSR1->LSR3->LSR4->LSR2->Egress

PBTS Function Details

The following PBTS functions are supported on the routter:

- Classify the Ingress traffic into different classes by creating rules using PBR configuration.
- Classify packets using DSCP/IP precedence for both IPv4 and IPv6 traffic.
- After classification, set the desired forward-class to each type of Ingress traffic.
- Define one or many MPLS-TE tunnels in the destination using Tunnel configuration.
- Associate the MPLS-TE tunnels to a specific forward-class under Tunnel configuration.
- Enable PBTS on the Ingress interface by applying the service policy that uses the configured classification rules.

The following list gives PBTS support information:

- PBTS is supported only on Ipv4/Ipv6 incoming traffic only.
- A maximum of eight forward-classes per destination prefix is supported.
- A maximum of 64 TE-tunnels within each forward class is supported.

- A maximum of 64 TE-tunnels can be configured on a given destination.
- Incoming labeled traffic is not supported.
- PBTS with L2VPN/L3VPN traffic is not supported.

PBTS Forward Class

A class-map is defined for various types of packets and these class-maps are associated with a forward-class. A class-map defines the matching criteria for classifying a particular type of traffic and a forward-class defines the forwarding path these packets should take.

After a class-map is associated with a forwarding-class in the policy map, all the packets that match the class-map are forwarded as defined in the policy-map. The egress traffic engineering (TE) tunnel interfaces that the packets should take for each forwarding-class is specified by associating the TE interface explicitly (or implicitly in case of default value) with the forward-group.

When the TE interfaces are associated with the forward-class, they can be exported to the routing protocol module using the **auto-route** command. This will then associate the route in the FIB database with these tunnels. If the TE interface is not explicitly associated with a forward-class, it gets associated with a default-class (0). All non-TE interfaces will be routed to the forwarding plane (with forward-class set to default-class) by the routing protocol.

UCMP Over MPLS-TE

In Equal Cost Multi Path (ECMP), you can load-balance routed traffic over multiple paths of the same cost. With Unequal-Cost Multipath (UCMP), you can load-balance traffic over multiple paths of varying costs.

Consider three forwarding links, with two 10 Gigabit Ethernet links and a 100 Gigabit Ethernet link, as shown in the image.

In such a scenario, the incoming traffic load is not equally distributed using ECMP. On the other hand, UCMP applies a weight to a path, and adds more forwarding instances to a path that has a higher weight (larger bandwidth). This results in an equal load distribution over paths of varying bandwidths (and costs).

Configuration Example

UCMP Configuration:

```
R1# configure
R1(config)# mpls traffic-eng
R1(config-mpls-te)# load-share unequal
R1(config-mpls-te)# commit
```

Tunnels Configuration:

```
R1(config)# interface tunnel-te 1
R1(config-if)# ipv4 unnumbered loopback 1
R1(config-if)# load-share 5
R1(config-if)# autoroute announce
R1(config-if-tunte-aa)# commit
R1(config-if-tunte-aa)# exit
R1(config-if)# destination 172.16.0.1
R1(config-if)# path-option 1 dynamic

R1(config)# interface tunnel-te 2
```

```
R1(config-if)# ipv4 address 192.168.0.1 255.255.255.0
R1(config-if)# load-share 6
R1(config-if)# autoroute announce
R1(config-if-tunte-aa)# commit
R1(config-if-tunte-aa)# exit
R1(config-if)# destination 172.16.0.1
R1(config-if)# path-option 1 dynamic
```

Associated Commands

- load-share
- load-share unequal
- show cef

Verification

Verify UCMP Configuration:

```
R1# show cef 172.16.0.1 detail

172.16.0.1/32, version 16, internal 0x1000001 0x0 (ptr 0x97de1a58) [1], 0x0 (0x97fa3728),
0xa20 (0x98fc00a8)
 Updated Jun 17 16:07:46.325
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x97e0ba08) reference count 3, flags 0x68, source lsd (5), 1 backups
                [3 type 4 flags 0x8401 (0x9849f728) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0x97fa3728, sh-ldi=0x9849f728]
  gateway array update type-time 1 Jun 17 16:07:46.325
 LDI Update time Jun 17 16:07:46.350
 LW-LDI-TS Jun 17 16:07:46.350
   via 172.16.0.1/32, tunnel-te1, 5 dependencies, weight 100, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x98e19380 0x98e192f0]
    next hop 172.16.0.1/32
    local adjacency
     local label 24001      labels imposed {ImplNull}
   via 172.16.0.1/32, tunnel-te2, 7 dependencies, weight 10, class 0 [flags 0x0]
    path-idx 1 NHID 0x0 [0x98e194a0 0x98e19410]
    next hop 172.16.0.1/32
    local adjacency
     local label 24001      labels imposed {ImplNull}

    Weight distribution:

    slot 0, weight 100, normalized_weight 10, class 0
    slot 1, weight 10, normalized_weight 1, class 0
    Load distribution: 0 0 0 0 0 0 0 0 0 0 1 (refcount 3)

    Hash  OK  Interface                 Address
    0     Y   tunnel-te1                point2point
    1     Y   tunnel-te1                point2point
    2     Y   tunnel-te1                point2point
    3     Y   tunnel-te1                point2point
    4     Y   tunnel-te1                point2point
    5     Y   tunnel-te1                point2point
    6     Y   tunnel-te1                point2point
    7     Y   tunnel-te1                point2point
    8     Y   tunnel-te1                point2point
    9     Y   tunnel-te1                point2point
    10    Y   tunnel-te2                point2point
```

Some sample output:

```
Router# show run formal mpls traffic-eng

mpls traffic-eng
mpls traffic-eng interface Bundle-Ether2
mpls traffic-eng interface Bundle-Ether3
..
mpls traffic-eng load-share unequal
mpls traffic-eng reoptimize 180
mpls traffic-eng signalling advertise explicit-null
mpls traffic-eng reoptimize timers delay path-protection 60

Router# show run formal interface tunnel-te400

interface tunnel-te400
interface tunnel-te400 description TE-STI-GRTMIABR5-GRTBUEBA3-BW-0
interface tunnel-te400 ipv4 unnumbered Loopback0
interface tunnel-te400 load-interval 30
interface tunnel-te400 signalled-name TE-STI-GRTMIABR5-GRTBUEBA3-BW-0
interface tunnel-te400 load-share 80
interface tunnel-te400 autoroute destination 94.142.100.214
interface tunnel-te400 destination 94.142.100.214
interface tunnel-te400 path-protection
interface tunnel-te400 path-option 1 explicit name BR5-BA3-0 protected-by 2
interface tunnel-te400 path-option 2 explicit name BW-BR5-1-VAP3-BA3-0
interface tunnel-te400 path-option 3 explicit name BW-BR5-VAP3-BA3-0

Router# show run formal interface tunnel-te406

interface tunnel-te406
interface tunnel-te406 description TE-STI-GRTMIABR5-GRTBUEBA3-BW-20
interface tunnel-te406 ipv4 unnumbered Loopback0
interface tunnel-te406 load-interval 30
interface tunnel-te406 signalled-name TE-STI-GRTMIABR5-GRTBUEBA3-BW-20
interface tunnel-te406 load-share 10
interface tunnel-te406 autoroute destination 94.142.100.214
interface tunnel-te406 destination 94.142.100.214
interface tunnel-te406 path-protection
interface tunnel-te406 path-option 1 explicit name BR5-1-BA3-0 protected-by 2
interface tunnel-te406 path-option 2 explicit name BW-BR5-VAP4-BA3-0

Router# show cef 94.142.100.214/32 det

94.142.100.214/32, version 25708656, attached, internal 0x4004081 0x0 (ptr 0x764ff1b0) [3],
 0x0 (0x7267d848), 0x440 (0x7d93b2b8)
 Updated Nov 19 08:02:26.545
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x72411528) reference count 3, flags 0xd0, source lsd (4), 1 backups
                [3 type 4 flags 0x10101 (0x7300d648) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0x7267d848, sh-ldi=0x7300d648]
   via tunnel-te400, 3 dependencies, weight 80, class 0 [flags 0x8]
    path-idx 0 NHID 0x0 [0x72082a40 0x72983b58]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te406, 3 dependencies, weight 10, class 0 [flags 0x8]
    path-idx 1 NHID 0x0 [0x7207b4ac 0x729886bc]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te410, 3 dependencies, weight 80, class 0 [flags 0x8]
    path-idx 2 NHID 0x0 [0x72085218 0x72985ee4]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te426, 3 dependencies, weight 10, class 0 [flags 0x8]
    path-idx 3 NHID 0x0 [0x7207d4b4 0x7297fecc]
```

```
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te427, 3 dependencies, weight 25, class 0 [flags 0x8]
    path-idx 4 NHID 0x0 [0x720802cc 0x7298726c]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te1089, 3 dependencies, weight 40, class 0 [flags 0x8]
    path-idx 5 NHID 0x0 [0x72081848 0x7298037c]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te1090, 3 dependencies, weight 60, class 0 [flags 0x8]
    path-idx 6 NHID 0x0 [0x7207d770 0x72987780]
    local adjacency
     local label 16440      labels imposed {ImplNull}
   via tunnel-te1099, 3 dependencies, weight 60, class 0 [flags 0x8]
    path-idx 7 NHID 0x0 [0x7207ed50 0x72981c7c]
    local adjacency
     local label 16440      labels imposed {ImplNull}

    Weight distribution:
    slot 0, weight 80, normalized_weight 7, class 0
    slot 1, weight 10, normalized_weight 1, class 0
    slot 2, weight 80, normalized_weight 7, class 0
    slot 3, weight 10, normalized_weight 1, class 0
    slot 4, weight 25, normalized_weight 1, class 0
    slot 5, weight 40, normalized_weight 3, class 0
    slot 6, weight 60, normalized_weight 5, class 0
    slot 7, weight 60, normalized_weight 5, class 0
Router# show cef 94.142.100.213/32 det

94.142.100.213/32, version 25708617, attached, internal 0x4004081 0x0 (ptr 0x771925c8) [3],
 0x0 (0x7267a594), 0x440 (0x7d93d364)
 Updated Nov 19 08:02:01.029
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x7240f638) reference count 3, flags 0xd0, source lsd (4), 1 backups
                [3 type 4 flags 0x10101 (0x73013360) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0x7267a594, sh-ldi=0x73013360]
   via tunnel-te220, 3 dependencies, weight 60, class 0 [flags 0x8]
    path-idx 0 NHID 0x0 [0x7207d838 0x72982af0]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te230, 3 dependencies, weight 60, class 0 [flags 0x8]
    path-idx 1 NHID 0x0 [0x7207d068 0x72986e20]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te236, 3 dependencies, weight 50, class 0 [flags 0x8]
    path-idx 2 NHID 0x0 [0x720830e4 0x7297f508]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te246, 3 dependencies, weight 100, class 0 [flags 0x8]
    path-idx 3 NHID 0x0 [0x7207a1ec 0x7298483c]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te221, 3 dependencies, weight 50, class 0 [flags 0x8]
    path-idx 4 NHID 0x0 [0x7207ea30 0x72982834]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te222, 3 dependencies, weight 25, class 0 [flags 0x8]
    path-idx 5 NHID 0x0 [0x72084a48 0x72989850]
    local adjacency
     local label 17561      labels imposed {ImplNull}
   via tunnel-te1091, 3 dependencies, weight 30, class 0 [flags 0x8]
    path-idx 6 NHID 0x0 [0x720851b4 0x729895f8]
    local adjacency
```

```
  local label 17561      labels imposed {ImplNull}
via tunnel-te342, 3 dependencies, weight 100, class 0 [flags 0x8]
 path-idx 7 NHID 0x0 [0x72085344 0x7298b024]
 local adjacency
  local label 17561      labels imposed {ImplNull}

 Weight distribution:
 slot 0, weight 60, normalized_weight 2, class 0
 slot 1, weight 60, normalized_weight 2, class 0
 slot 2, weight 50, normalized_weight 2, class 0
 slot 3, weight 100, normalized_weight 4, class 0
 slot 4, weight 50, normalized_weight 2, class 0
 slot 5, weight 25, normalized_weight 1, class 0
 slot 6, weight 30, normalized_weight 1, class 0
 slot 7, weight 100, normalized_weight 4, class 0

 Load distribution: 0 0 1 1 2 2 3 3 3 3 4 4 5 6 7 7 7 7 (refcount 3)

 Hash  OK  Interface                 Address
 0     Y   tunnel-te220             point2point
 1     Y   tunnel-te220             point2point
 2     Y   tunnel-te230             point2point
 3     Y   tunnel-te230             point2point
 4     Y   tunnel-te236             point2point
 5     Y   tunnel-te236             point2point
 6     Y   tunnel-te246             point2point
 7     Y   tunnel-te246             point2point
 8     Y   tunnel-te246             point2point
 9     Y   tunnel-te246             point2point
 10    Y   tunnel-te221             point2point
 11    Y   tunnel-te221             point2point
 12    Y   tunnel-te222             point2point
 13    Y   tunnel-te1091            point2point
 14    Y   tunnel-te342             point2point
 15    Y   tunnel-te342             point2point
 16    Y   tunnel-te342             point2point
 17    Y   tunnel-te342             point2point
```

Configuring Performance Measurement

Network performance metrics such as packet loss, delay, delay variation, and bandwidth utilization is a critical measure for traffic engineering (TE) in service provider networks. These network performance metrics provide network operators information about the performance characteristics of their networks for performance evaluation and helps to ensure compliance with service level agreements. The service-level agreements (SLAs) of service providers depend on the ability to measure and monitor these network performance metrics. Network operators can use performance measurement (PM) feature to monitor the network metrics for links as well as end-to-end TE label switched paths (LSPs).

Path Calculation Metric Type

To configure the metric type to be used for path calculation for a given tunnel, use the **path-selection metric** command in either the MPLS-TE configuration mode or under the tunnel interface configuration mode.

The metric type specified per interface takes the highest priority, followed by the MPLS-TE global metric type.

Note If the delay metric is configured, CSPF finds a path with optimized*minimum* link delay metric. See the *Configuring Performance Measurement* chapter in the Segment Routing Configuration Guide for information on configuring interface performance delay measurement.

Configuration Example

The following example shows how to set the path-selection metric to use the IGP metric under a specific tunnel interface:

```
Router# configure
Router(config)# interface tunnel-te 1
Router(config-if)# path-selection metric igp
Router(config-if)# commit
```

The following example shows how to set the path-selection metric to use the delay metric under the MPLS-TE configuration mode:

```
Router# configure
Router(config)# mpls traffic-eng
Router(config-mpls-te)# path-selection metric delay
Router(config-mpls-te)# commit
```

Path-Selection Delay Limit

Apply the **path-selection delay-limit** configuration to set the upper limit on the path aggregate delay when computing paths for MPLS-TE LSPs. After you configure the **path-selection delay-limit** value, if the sum of minimum-delay metric from all links that are traversed by the path exceeds the specified delay-limit, CSPF will not return any path. The periodic path verification checks if the delay-limit is crossed.

The **path-selection delay-limit** value can be configured at the global MPLS-TE, per-interface tunnel, and per path-option attribute set. The path-selection delay-limit per path-option attribute set takes the highest priority, followed by per-interface, and then the MPLS-TE global path-selection delay-limit values.

The delay limit range is a value from 1 to 4294967295 microseconds.

Note See the *Configuring Performance Measurement* chapter in the Segment Routing Configuration Guide for information on configuring interface performance delay measurement.

Configuration Example

The following example shows how to set the path-selection delay limit under a specific tunnel interface:

```
Router# configure
Router(config)# interface tunnel-te2000
Router(config-if)# path-selection metric delay
Router(config-if)# path-selection delay-limit 200
Router(config-if)# commit
```

The following example shows how to set the path-selection delay limit under a path-option attribute set:

```
Router# configure
Router(config)# mpls traffic-eng
Router(config-mpls-te)# attribute-set path-option test
Router(config-te-attribute-set)# path-selection delay-limit 300
Router(config-te-attribute-set)# root
```

```
Router(config)# interface tunnel-te1000
Router(config-if)# path-option 10 dynamic attribute-set test
Router(config-if)# commit
```

The following example shows how to set the path-selection delay limit under the global MPLS-TE configuration mode:

```
Router# configure
Router(config)# mpls traffic-eng
Router(config-mpls-te)# path-selection metric delay
Router(config-mpls-te)# path-selection delay-limit 150
Router(config-mpls-te)# commit
```