

MULTI-TOPOLOGY ROUTING

Table of Contents

Executive Summary	3
Introduction	3
Directing Traffic Types	3
Multi-Topology RIBs	4
Direct and Local Routes	5
Static Routes	5
OSPF.....	5
BGP.....	7
Resolving BGP Next Hops	9
Multi-Topology Forwarding Plane.....	10
Case Study: Choosing a Topology Path Based on an Application	11
Configuring the Routing Control Plane.....	12
Configuring the Forwarding Plane	18
Example Output Showing How Traffic Traverses the Network.....	20
Case Study: Choosing a Topology Path Based on a Multicast Source	21
Conclusion	28
Acronyms	29
Appendix A: Configurations for the Case Study on Choosing a Topology Path Based on an Application	29
PE1 Router Configurations	29
PE2 Router Configurations	32
P2 Router Configurations.....	35
CE1 Router Configurations	38
Appendix B: Configurations for the Case Study on Choosing a Topology Path Based on a Multicast Source.....	40
PE1 Router Configurations	40
PE2 Router Configurations	44
P1 Router Configurations.....	47
CE1 Router Configurations	49
For More Information	51
About Juniper Networks.....	51

Table of Figures

Figure 1: Voice and video routing topologies enabled on a subset of links.....	3
Figure 2: Multi-topology OSPF for designating links belonging to voice and video services	11
Figure 3: IBGP core links configured to prefer specified routing topologies	21
Figure 4: Two multicast trees traversing two different paths via routing topologies	25

Executive Summary

This white paper provides a basic description of how service providers and enterprises can use multi-topology routing (MTR) to engineer traffic flow across a network. Within this framework, the paper describes how MTR works in Juniper Networks® Junos® operating system, including direct and static routes, OSPF, and BGP. The paper also describes how to configure an MTR network for unicast and multicast IP, using Junos OS.

Two case studies are explored in which different components of MTR are used. The first case is basic unicast IP, where an IBGP core runs on top of OSPF. The second case describes how a PIM is used, in conjunction with MTR OSPF and BGP, to direct multicast traffic over particular paths based on traffic characteristics.

Introduction

In a network carrying multiple traffic types, you often need to direct different types of application traffic over multiple links depending on their link characteristics. For example, voice traffic requires links that are less likely to incur latency, jitter, or packet loss; file traffic, on the other hand, requires links that have large amounts of available bandwidth.

MTR is a way to direct traffic to follow specified paths. You can use MTR to extend a traditional MPLS network into a segment where only IP forwarding is supported. With MTR, each traffic type is handled in its own conceptually incongruent topology, and yet runs on top of the same underlying network; you can configure separate topologies to share the same network links as needed.

MTR uses a combination of control plane (routing) and forwarding plane (firewall filters). Each topology uses the unified control plane to make routing decisions for traffic associated with that topology. As well, each topology has a separate forwarding information base (FIB) and in effect, a dedicated forwarding plane for each topology. This forwarding plane not only directs traffic using its own FIB, but also simultaneously handles sophisticated functionality, such as queuing for class of service, that can be applied to a topology. As traffic enters the router, fields within a packet determine to which topology the traffic belongs.

Directing Traffic Types

One way to manage traffic flow is to group links into specific routing topologies based on application requirements. Each routing topology can be thought of as a set of contiguous links. MTR provides a way for you to manage each set of links uniquely by directing traffic types to flow over specified links. This solution uses a combination of routing (control plane) and firewall filtering (forwarding plane) configurations.

Figure 1 shows a network with two topologies configured: voice (dotted lines) and video (dashed lines). Note there is also a default routing topology (solid line).

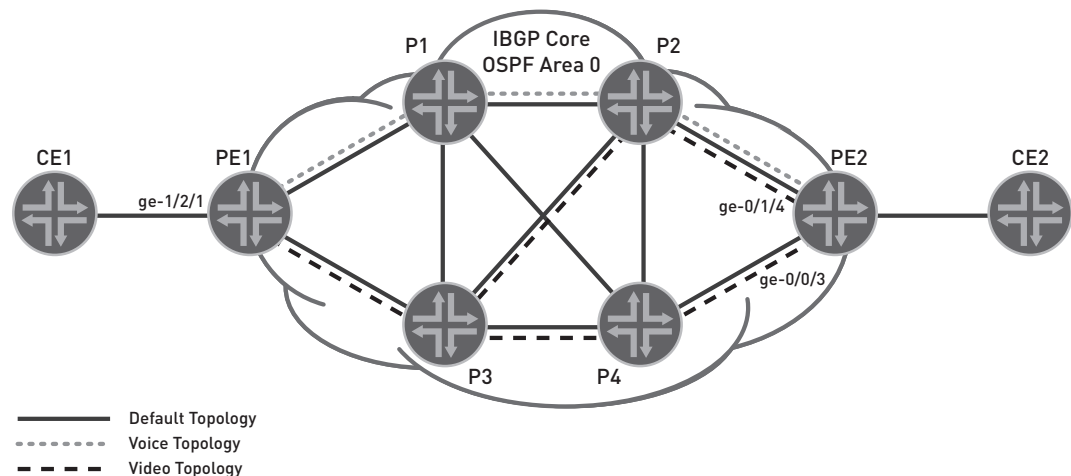


Figure 1: Voice and video routing topologies enabled on a subset of links

You can configure MTR for BGP, OSPF, and static routes. When a routing topology is created, it has its own routing information base (RIB), which in turn is used to derive its own FIB.

Packet forwarding uses firewall filters to examine packets as they enter the router over an interface. These filters determine whether a specific topology or the default FIB should be used to make packet forwarding decisions. If applicable, firewall filters evaluate packet attributes, such as destination IP address, DSCPs, or next-level protocol headers, to determine which topology to use. In fact, any item in a packet that is recognized by a firewall filter can be used to direct the packet next-hop lookup to use a particular topology. Once the packet is directed to use a topology, the destination IP address must be in the topology FIB; otherwise, the packet is dropped.

Topologies are configured under `routing-options`.

Example: This configuration shows voice and video topologies, which will enable you to use these topologies with OSPF and BGP routing protocols, as well as with static routes.

```
.....  
routing-options {  
    topologies {  
        family inet {  
            topology voice;  
            topology video;  
        }  
    }  
}
```

```
.....
```

Note that the names `voice` and `video` are local to the router. The actual names are not propagated beyond this router. However, for management purposes, a consistent naming scheme across routers in a multi-topology environment is convenient.

Multi-Topology RIBs

Routing topologies have their own RIBs, similar to any other RIB created by default or by a `rib-group` configuration with a few differences. The RIB name indicates that the RIB is associated with a topology by prepending a ":" to the name. For example, a routing topology named `voice` has a RIB named `:voice.inet.0`.

When routing topologies are configured under `routing-options`, a new RIB for each topology is created.

Example: This output shows RIBs `:voice.inet.0` and `:video.inet.0` are created after configuring topologies `voice` and `video` under the `routing-options` stanza.

```
.....  
rtr> show route summary  
...  
:voice.inet.0: 8 destinations, 8 routes (7 active, 0 holddown, 1 hidden)  
    Direct:      4 routes,      3 active  
    Local:       4 routes,      4 active  
  
:video.inet.0: 8 destinations, 8 routes (7 active, 0 holddown, 1 hidden)  
    Direct:      4 routes,      3 active  
    Local:       4 routes,      4 active  
...  
.....
```

Direct and Local Routes

When a topology RIB is created, it automatically copies all direct and local routes in the default RIB (`inet.0`) to the new topology RIB.

Example: This output shows direct and local routes automatically populating the topology RIB `:video.inet.0` when this topology is created under `routing-options`.

```
.....
rtr> show route table :video.inet.0
:video.inet.0: 6 destinations, 6 routes (5 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.0/24      *[Direct/0] 00:02:29
                > via fe-0/2/0.0
1.1.1.1/32     *[Local/0] 00:02:29
                Local via fe-0/2/0.0
10.255.165.93/32 *[Direct/0] 00:04:34
                > via lo0.0
192.168.164.0/22 *[Direct/0] 00:04:34
                > via fxp0.0
192.168.165.93/32 *[Local/0] 00:02:29
                Local via fxp0.0
.....
```

Static Routes

You can add static routes into a topology RIB by configuring static routes under a particular topology RIB.

Example: This configuration shows a static route being added to topology RIB `:voice.inet.0`.

```
.....
routing-options {
  rib :voice.inet.0 {
    static {
      route 2.2.2.0/24 next-hop 11.19.1.2;
    }
  }
}
.....
```

Example: This output shows static route `2.2.2.0/24` was placed in the topology RIB `:voice.inet.0`.

```
.....
rtr> show route table :voice.inet.0 protocol static

:voice.inet.0: 29 destinations, 30 routes (28 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

2.2.2.0/24     *[Static/5] 00:00:17
                > to 11.19.1.2 via fe-0/2/1.0
.....
```

OSPF

Junos OS supports routing topologies with OSPF; the implementation of MTR OSPF is based on RFC 4915 (Multi-Topology [MT] Routing in OSPF). As previously noted, topology names are local to the router. An OSPF multi-topology identification (MT-ID) is a number chosen to correspond with a specific routing topology and is configured on the router. The MT-ID must be consistent on all routers participating in the same MTR OSPF topology. The MT-ID identifies to which topology the OSPF LSAs belong; the value ranges from 32 to 127. This configured MT-ID value is copied directly into the previously unused ToS field of the LSA.

MTR is configured under the OSPF protocols stanza, and OSPF interfaces are configured to belong to one or more topologies. The Router-LSA originated and flooded by the router includes all relevant topology information for specific interfaces, such as MT-ID and metric. If MTR is not configured on an OSPF interface, then the Router-LSA does not include any topology information for that interface.

OSPF neighbors may or may not support MTR OSPF. That is, a particular link is not used to calculate OSPF routes for a topology unless routers at both ends of the link announce that link as part of the topology. If MTR OSPF is not supported in neighboring OSPF routers or is not configured to do so, then topology information in LSAs received by the neighbor is ignored.

Example: This PE2 configuration (Figure 1) shows MTR configured on two OSPF interfaces; the topology name is video and the MT-ID is 52.

```
.....
protocols {
  ospf {
    topology video topology-id 52;
    area 0.0.0.0 {
      interface ge-0/1/4.0 {
        topology video;
      }
      interface ge-0/0/3.0 {
        topology video;
      }
    }
  }
}
.....
```

Example: This output shows the video topology was added to OSPF interfaces ge-0/0/3 and ge-0/1/4.

```
.....
rtr-PE2> show ospf interface ge-0/0/3.0 detail
Interface      State   Area          DR ID          BDR ID          Nbrs
ge-0/0/3.0     BDR    0.0.0.0       10.255.165.99 10.255.164.203  1
  Type: LAN, Address: 11.19.27.1, Mask: 255.255.255.0, MTU: 1500, Cost: 1
  DR addr: 11.19.27.2, BDR addr: 11.19.27.1, Priority: 128, Adj count: 1
  Hello: 10, Dead: 40, ReXmit: 5, Not Stub
  Auth type: None
  Topology default (ID 0) -> Cost: 1
  Topology video (ID 52) -> Cost: 1 <- This OSPF interface shows topology video
.....

rtr-PE2> show ospf interface ge-0/1/4.0 detail
Interface      State   Area          DR ID          BDR ID          Nbrs
ge-0/1/4.0     BDR    0.0.0.0       10.255.165.113 10.255.164.203  1
  Type: LAN, Address: 11.19.25.1, Mask: 255.255.255.0, MTU: 1500, Cost: 1
  DR addr: 11.19.25.2, BDR addr: 11.19.25.1, Priority: 128, Adj count: 1
  Hello: 10, Dead: 40, ReXmit: 5, Not Stub
  Auth type: None
  Topology default (ID 0) -> Cost: 1
  Topology video (ID 52) -> Cost: 1 <- This OSPF interface shows topology video
.....
```

You can also independently configure MT-OSPF interface metric values for different topologies, thus allowing topologies to be incongruent with each other.

Example: This configuration shows the topology video with metrics configured on both MT-OSPF interfaces.

```

.....
protocols {
  ospf {
    topology video topology-id 52;
    area 0.0.0.0 {
      interface ge-0/1/4.0 {
        topology video metric 10; <- MT-OSPF interface metric of 10
      }
      interface ge-0/0/3.0 {
        topology video metric 5; <- MT-OSPF interface metric of 5
      }
    }
  }
}
.....

```

Example: The output for the OSPF interfaces now has a cost associated with the topology interfaces.

```

.....
rtr-PE2> show ospf interface ge-0/0/3.0 detail
Interface      State  Area          DR ID          BDR ID          Nbrs
ge-0/0/3.0     BDR    0.0.0.0       10.255.165.99  10.255.164.203  1
Type: LAN, Address: 11.19.27.1, Mask: 255.255.255.0, MTU: 1500, Cost: 1
DR addr: 11.19.27.2, BDR addr: 11.19.27.1, Priority: 128, Adj count: 1
Hello: 10, Dead: 40, ReXmit: 5, Not Stub
Auth type: None
Topology default (ID 0) -> Cost: 1
Topology video (ID 52) -> Cost: 10 <- The cost is now 10

rtr-PE2> show ospf interface ge-0/1/4.0 detail
Interface      State  Area          DR ID          BDR ID          Nbrs
ge-0/1/4.0     BDR    0.0.0.0       10.255.165.113 10.255.164.203  1
Type: LAN, Address: 11.19.25.1, Mask: 255.255.255.0, MTU: 1500, Cost: 1
DR addr: 11.19.25.2, BDR addr: 11.19.25.1, Priority: 128, Adj count: 1
Hello: 10, Dead: 40, ReXmit: 5, Not Stub
Auth type: None
Topology default (ID 0) -> Cost: 1
Topology video (ID 52) -> Cost: 5 <- The cost is now 5
.....

```

BGP

Unlike OSPF, there is no RFC specification for multi-topology BGP. In Junos OS, multi-topology support for BGP is based on the community value in a BGP route. A configuration under BGP is used to determine the association between a topology and one or more community values. BGP routes populate topology RIBs by applying configurations under the BGP protocols stanza. A routing topology name and BGP community value are configured. Subsequent BGP updates arriving that have a matching community value are replicated in the associated topology RIB. You decide which BGP community values are associated with a given topology.

Example: This configuration indicates that BGP updates received with community value `target:40:40` should be added into topology RIB `:voice.inet.0` (in addition to the default RIB `inet.0`).

```

.....
protocols {
  bgp {
    group ibgp {
      family inet {
        unicast {
          topology voice {
            community target:40:40;
          }
        }
      }
    }
  }
}
.....

```

Example: This output from PE1 (Figure 1) shows BGP route `11.19.130.0/24` with community value `target:40:40`. Since it matches the criteria for the voice topology, it is added to both the default and voice topology RIBs (`inet.0` and `:voice.inet.0`). PE1 learns the route from CE1 via EBGp and then further injects it into IBGP.

```

.....
rtr-PE1 > show route 11.19.130.0/24 detail

inet.0: 48 destinations, 48 routes (47 active, 0 holddown, 1 hidden)
11.19.130.0/24 (1 entry, 1 announced)
  *BGP Preference: 170/-101
    Next hop type: Router, Next hop index: 550
    Next-hop reference count: 18
    Source: 11.19.30.2
    Next hop: 11.19.30.2 via ge-1/2/1.0, selected
    State: <Active Ext>
    Local AS: 100 Peer AS: 101
    Age: 8:14
    Task: BGP_101.11.19.30.2+56219
    Announcement bits (3): 0-KRT 4-BGP RT Background 5-Resolve tree 3
    AS path: 101 I
    Communities: target:40:40
    Localpref: 100
    Router ID: 10.255.165.113
    Secondary Tables: :voice.inet.0

:voice.inet.0: 27 destinations, 27 routes (26 active, 0 holddown, 1 hidden)
11.19.130.0/24 (1 entry, 1 announced)
  *BGP Preference: 170/-101
    Next hop type: Router, Next hop index: 550
    Next-hop reference count: 18
    Source: 11.19.30.2
    Next hop: 11.19.30.2 via ge-1/2/1.0, selected
    State: <Secondary Active IndepResolution Ext>
    Local AS: 100 Peer AS: 101
    Age: 8:14
    Task: BGP_101.11.19.30.2+56219
    Announcement bits (2): 0-Resolve tree 1 2-KRT
    AS path: 101 I
    Communities: target:40:40
    Localpref: 100
    Router ID: 10.255.165.113
    Primary Routing Table inet.0
.....

```

Resolving BGP Next Hops

A typical IBGP core has BGP routes with protocol next hops that resolve using the underlying IGP routes. IBGP routes in a topology RIB have protocol next-hop IP addresses, and by default, the same topology RIB is used to look up and resolve the protocol next-hop IP address to a forwarding next hop.

Example: This output from PE2 (Figure 1) shows the same BGP route as seen in the previous example:

11.19.130.0/24. Note that the route is being shown from a different perspective, that is, from PE2 as an IBGP route.

Similarly, this IBGP route is added to both `inet.0` and `:voice.inet.0` on PE2. However, while each route has the same protocol next hop, each route has a different forwarding next hop (`ge-0/0/3.0` vs. `ge-0/1/4.0`). The reason for this difference is when the protocol next-hop IP address `10.255.165.93` is resolved, it uses the corresponding RIB table (`inet.0` vs. `:voice.inet.0`) to look up the protocol next hop.

```

.....
rtr-PE2> show route 11.19.130.0/24 detail
inet.0: 48 destinations, 48 routes (47 active, 0 holddown, 1 hidden)
11.19.130.0/24 (1 entry, 1 announced)
    *BGP    Preference: 170/-101
            Next hop type: Indirect
            Next-hop reference count: 9
            Source: 10.255.165.93
            Next hop type: Router, Next hop index: 605
            Next hop: 11.19.27.2 via ge-0/0/3.0, selected <- Forwarding next hop
            Protocol next hop: 10.255.165.93 <- Protocol next hop
            Indirect next hop: 8d58000 1048575
            State: <Active Int Ext>
            Local AS: 100 Peer AS: 100
            Age: 11:19 Metric2: 29
            Task: BGP_100.10.255.165.93+179
            Announcement bits (3): 0-KRT 4-BGP RT Background 5-Resolve tree 2
            AS path: 101 I
            Communities: target:40:40
            Accepted
            Localpref: 100
            Router ID: 10.255.165.93
            Secondary Tables: :voice.inet.0
:voice.inet.0: 21 destinations, 21 routes (20 active, 0 holddown, 1 hidden)
11.19.130.0/24 (1 entry, 1 announced)
    *BGP    Preference: 170/-101
            Next hop type: Indirect
            Next-hop reference count: 6
            Source: 10.255.165.93
            Next hop type: Router, Next hop index: 562
            Next hop: 11.19.25.2 via ge-0/1/4.0, selected <- Forwarding next hop
            Protocol next hop: 10.255.165.93 <- Protocol next hop
            Indirect next hop: 8d58140 1048577
            State: <Secondary Active IndepResolution Int Ext>
            Local AS: 100 Peer AS: 100
            Age: 11:06 Metric2: 30
            Task: BGP_100.10.255.165.93+179
            Announcement bits (2): 0-Resolve tree 4 1-KRT
            AS path: 101 I
            Communities: target:40:40
            Accepted
            Localpref: 100
            Router ID: 10.255.165.93
            Primary Routing Table inet.0
.....

```

Example: This output from PE2 (Figure 1) shows the protocol next hop of 11.19.130.0/24, which is IP address 10.255.165.93, thus further demonstrating how IBGP route 11.19.130.0/24 resolves its protocol next hop. The forwarding next hops of 10.255.165.93 match the IBGP forwarding next hops of route 11.19.130.0/24 as shown in the previous example. Observe here that the IP address 10.255.165.93 is also in RIB :video.inet.0. This address is the loopback address of PE1, and as such, resides in all three RIBs.

This example also shows how traffic entering PE2 destined to 11.19.130.0/24 exits out different interfaces depending on its associated topology. The actual traffic is marked in such a way that a firewall filter can direct the traffic to use a particular topology RIB.

```

.....
rtr-PE2> show route 10.255.165.93

inet.0: 48 destinations, 48 routes (47 active, 0 holddown, 1 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both
10.255.165.93/32    *[OSPF/10] 00:25:10, metric 29
                  > to 11.19.27.2 via ge-0/0/3.0    <- Forwarding next hop

:voice.inet.0: 21 destinations, 21 routes (20 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.255.165.93/32    *[OSPF/10] 00:25:03, metric 30
                  > to 11.19.25.2 via ge-0/1/4.0    <- Forwarding next hop

:video.inet.0: 20 destinations, 20 routes (19 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.255.165.93/32    *[OSPF/10] 00:25:10, metric 29
                  > to 11.19.27.2 via ge-0/0/3.0
.....

```

You can also indicate up to two RIBs to perform the protocol next-hop resolution by using a configuration under the `routing-options resolution`. If two RIBs are indicated and the protocol next-hop IP address resides in both RIBs, normal tie-breaking rules, such as route preference or route metric, are applied to choose which RIB entry to use.

Multi-Topology Forwarding Plane

Once routing topologies are configured, traffic must go through a firewall filter to make use of routing topology FIBs.

For basic routing topologies, where traffic is first entering the core network, apply an input firewall filter to the ingress interface. Additionally, add firewall filters to interfaces where MT-OSPF is configured. All routers must use the same firewall filter to associate packets with a topology to ensure consistent forwarding and to avoid routing loops or packet loss.

The forwarding plane handles traffic as it enters the router and exits out a particular interface. To inspect traffic and use a specified topology FIB to perform next-hop lookups, configure an input firewall filter on each interface where routing topology support is desired. Use a regular firewall filter to identify packet characteristics. In general, for application-level differentiation, it is convenient to use DSCPs. When there is a firewall filter match, the firewall instructs the route lookup to use a particular topology FIB.

Example: This configuration shows an example firewall filter. Packet attributes are identified in the `from` clause, followed by a `then` clause indicating the topology FIB for use in forwarding next-hop lookups. This configuration notifies the router which traffic uses a routing topology FIB and which traffic uses the default FIB. Note that the last term, which is named `default`, specifies the use of the default FIB.

```

firewall {
  family inet {
    filter ef_path {
      term ef {
        from {
          forwarding-class expedited-forwarding;
        }
        then topology voice;
      }
    }
    term video {
      from {
        source-address {
          11.19.132.0/24;
        }
      }
      then topology video;
    }
    term default {
      then accept;
    }
  }
}

```

Case Study: Choosing a Topology Path Based on an Application

In this case study, assume the network is running OSPF and IBGP in the core, but not MPLS. Even though you do not have traffic engineering, you still want to configure voice traffic to use one set of links and video traffic to use a different set of links. This traffic may or may not be destined for the same IP address. In some cases, both applications traverse the same link.

The solution uses MTR-based OSPF and BGP, along with firewall filters, to direct different traffic types over designated links. The routers use a fairly similar set of configurations, which reduces complexity and improves network management. For a complete set of configurations for PE1, PE2, P2, and CE1, see *Appendix A*.

The OSPF topologies are defined to support each service offering over the OSPF area. The links of a topology must be contiguous, consistent with a typical OSPF area. IBGP routes in each routing topology automatically use the associated OSPF topology RIB for protocol next-hop route resolution; no special route resolution configurations are required.

In this solution, multiple topologies can be configured over the same link. However, traffic in each application service class cannot traverse links unless they are configured for the topology designated for that service. Figure 2 shows a simplified diagram of this case. Contiguous paths for routing the voice topology are shown with dotted lines, and paths for routing the video topology are shown with dashed lines.

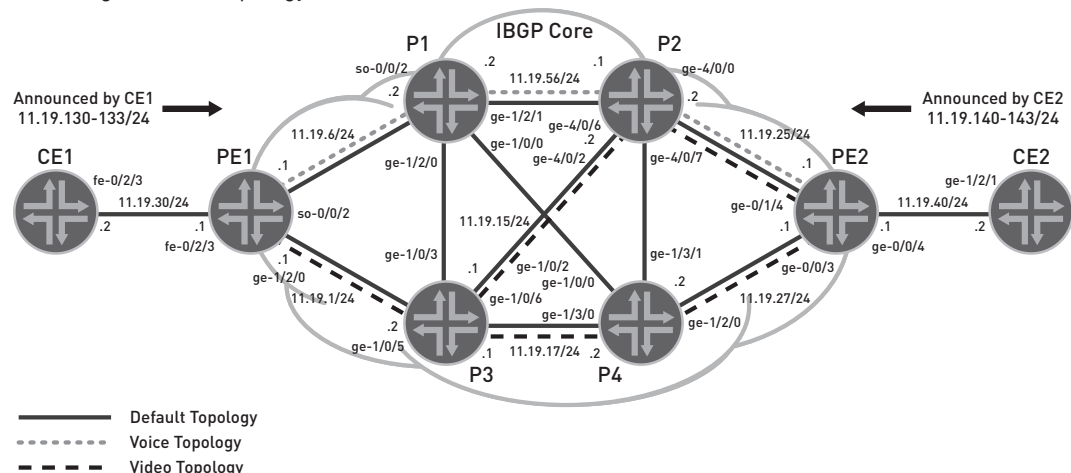


Figure 2: Multi-topology OSPF for designating links belonging to voice and video services

Configuring the Routing Control Plane

The routing control plane uses BGP communities to indicate traffic classes and update associated topology RIBs. The underlying IGP MT-OSPF uses topology RIBs and FIBs for protocol and forwarding next-hop lookups. The first task is to create voice and video topologies under `routing-options` on all core PE and P routers.

```
.....
routing-options {
  topologies {
    family inet {
      topology voice;
      topology video;
    }
  }
}
.....
```

The next task is to configure MT-OSPF. Configure voice and video topologies on PE and P routers. Apply MT-OSPF designations only on desired interfaces as shown in Figure 2.

- Apply the voice topology to OSPF on PE1, PE2, P1, and P2.
- Apply the video topology to OSPF on PE1, PE2, and P2-P4.

Here, P2 has voice topology configured on OSPF interfaces facing PE2 and P1. Similarly, P2 has video topology configured on OSPF interfaces facing PE2 and P3.

Once a topology ID is configured under OSPF, the topology is automatically enabled on all interfaces under OSPF. To disable a topology or add a metric, you must add an explicit configuration.

Example: This configuration shows how to enable and disable MT-OSPF on particular interfaces, in this case P2. For readability purposes, each topology is configured under each desired OSPF interface even though this default behavior occurs when the topology ID is configured. Note that the link between P2 and PE2 has a higher metric value configured on the video topology so that video traffic prefers the P4-PE2 link.

```
.....
protocols {
  ospf {
    topology voice topology-id 126;
    topology video topology-id 52;
    area 0.0.0.0 {
      interface ge-4/0/6.0 {
        description "P2 <-> P1";
        metric 10;
        topology voice;
        topology video disable;
      }
      interface ge-4/0/2.0 {
        description "P2 <-> P3";
        metric 10;
        topology voice disable;
        topology video metric 20;
      }
      interface ge-4/0/7.0 {
        description "P2 <-> P4";
        metric 10;
        topology voice disable;
        topology video disable;
      }
      interface ge-4/0/0.0 {
        description "P2 <-> PE2";
        metric 10;
        topology voice;
        topology video metric 200;
      }
    }
  }
}
.....
```

```

    }
    interface lo0.0 {
        passive;
    }
}
}

```

Example: This P2 output shows OSPF neighbor PE2 (11.19.25.1), where MT-OSPF default, video, and voice are seen as MT-OSPF participants. The `Bidirectional` flag shows you that the neighbor is configured using the same MT-OSPF ID.

```

rtr-P2> show ospf neighbor 11.19.25.1 extensive
Address      Interface      State   ID           Pri  Dead
11.19.25.1   ge-4/0/0.0     Full   10.255.164.203 128  39
Area 0.0.0.0, opt 0x42, DR 11.19.25.1, BDR 11.19.25.2
Up 00:11:50, adjacent 00:11:50
Topology default (ID 0) -> Bidirectional
Topology video (ID 52) -> Bidirectional
Topology voice (ID 126) -> Bidirectional

```

Example: This P2 output shows the MT-OSPF status with neighbor P1. Default and voice topologies are enabled.

```

rtr-P2> show ospf neighbor 11.19.56.2 extensive
Address      Interface      State   ID           Pri  Dead
11.19.56.2   ge-4/0/6.0     Full   10.255.165.95 128  37
Area 0.0.0.0, opt 0x42, DR 11.19.56.2, BDR 11.19.56.1
Up 00:33:51, adjacent 00:33:51
Topology default (ID 0) -> Bidirectional
Topology voice (ID 126) -> Bidirectional

```

Example: This P2 output shows the Router-LSA originated by PE2. The LSA shows links where video and voice topologies are enabled (in addition to the default topology).

```

rtr-P2> show ospf database lsa-id 10.255.164.203 extensive

OSPF link state database, Area 0.0.0.0
Type      ID           Adv Rtr      Seq         Age  Opt  Cksum  Len
Router    10.255.164.203 10.255.164.203 0x80000009 954  0x22 0xc70b 80
bits 0x0, link count 3
id 11.19.27.2, data 11.19.27.1, Type Transit (2)
Topology count: 1, Default metric: 10
Topology video (ID 52) -> Metric: 10      <- Link to P4 only has video enabled
id 11.19.25.1, data 11.19.25.1, Type Transit (2)
Topology count: 2, Default metric: 10
Topology video (ID 52) -> Metric: 200     <- Link to P2 has video topology enabled
Topology voice (ID 126) -> Metric: 10     <- Link to P2 also has voice topology enabled
id 10.255.164.203, data 255.255.255.255, Type Stub (3)
Topology count: 2, Default metric: 0
Topology video (ID 52) -> Metric: 0
Topology voice (ID 126) -> Metric: 0
Aging timer 00:44:06
Installed 00:15:53 ago, expires in 00:44:06, sent 00:15:53 ago
Last changed 00:15:53 ago, Change count: 3

```

Example: This P2 output shows an OSPF route 10.255.164.203 (originated by PE2) belonging to all three topologies: default, voice, and video. Due to different metrics on each topology, the paths for voice and video are different. The default and voice paths are one hop away to PE2, and the video path is via P3.

```

.....
rtr-P2> show route 10.255.164.203 terse

inet.0: 48 destinations, 48 routes (47 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O  10      10                >11.19.25.1   <- One hop to PE2
...
:voice.inet.0: 22 destinations, 22 routes (21 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O  10      10                >11.19.25.1   <- One hop to PE2

:video.inet.0: 22 destinations, 22 routes (21 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O  10      39                >11.19.15.1   <- Via P3 to PE2
.....

```

The third task is to configure routing topologies under BGP on the PE routers.

Example: In this configuration you are designating which BGP communities relate to which topologies.

- The BGP community `target:40:40` designates voice.
- BGP community `target:50:50` designates video.
- BGP routes with community `target:40:40` are copied into RIB `:voice.inet.0`.
- BGP routes with community `target:50:50` are copied into RIB `:video.inet.0`.

```

.....
protocols {
  bgp {
    group ibgp {
      family inet {
        unicast {
          topology voice {
            community target:40:40;
          }
          topology video {
            community target:50:50;
          }
        }
      }
    }
  }
}
.....

```

BGP routing updates arrive at the PE router with a BGP community value already assigned.

Example: This configuration shows a CE router wherein community values are added to BGP updates by using an export statement and a policy. BGP updates voice and BGP updates video are then announced to a PE router using EBGP. Note that these are basic BGP configurations that do not require MTR.

```

.....
protocols {
  bgp {
    group ebgp {
      export set_community;
    }
  }
}
policy-options {
  policy-statement set_community {
    term a {
      from {
        route-filter 11.19.130.0/24 exact;
        route-filter 11.19.131.0/24 exact;
      }
      then {
        community add voice;
        accept;
      }
    }
    term b {
      from {
        route-filter 11.19.132.0/24 exact;
        route-filter 11.19.133.0/24 exact;
      }
      then {
        community add video;
        accept;
      }
    }
    term default {
      then accept;
    }
  }
  community voice members target:40:40;
  community video members target:50:50;
}
.....

```

Example: This PE1 output shows a BGP route in inet.0 with community target:40:40:.

```

.....
rtr-PE1> show route 11.19.140 extensive table inet.0

inet.0: 48 destinations, 49 routes (47 active, 0 holddown, 1 hidden)
11.19.140.0/24 (1 entry, 1 announced)
TSI:
KRT in-kernel 11.19.140.0/24 -> {indirect(262142)}
Page 0 idx 0 Type 1 val 8ab6690
  Nexthop: Self
  AS path: [100] 102 I
  Communities: target:40:40
Path 11.19.140.0 from 10.255.164.203 Vector len 4. Val: 0
  *BGP Preference: 170/-101
  Next hop type: Indirect
  Next-hop reference count: 9
  Source: 10.255.164.203
.....

```

```

Next hop type: Router, Next hop index: 552
Next hop: 11.19.1.2 via ge-1/2/0.0, selected
Protocol next hop: 10.255.164.203
Indirect next hop: 8bad000 262142
State: <Active Int Ext>
Local AS: 100 Peer AS: 100
Age: 2:28:04 Metric2: 12
Task: BGP_100.10.255.164.203+63358
Announcement bits (3): 0-KRT 4-BGP RT Background 5-Resolve tree 3
AS path: 102 I
Communities: target:40:40
Localpref: 100
Router ID: 10.255.164.203
Secondary Tables: :voice.inet.0
Indirect next hops: 1
    Protocol next hop: 10.255.164.203 Metric: 12
    Indirect next hop: 8bad000 262142
    Indirect path forwarding next hops: 1
        Next hop type: Router
        Next hop: 11.19.1.2 via ge-1/2/0.0
10.255.164.203/32 Originating RIB: inet.0
Metric: 12 Node path count: 1
Forwarding nexthops: 1
    Nexthop: 11.19.1.2 via ge-1/2/0.0

```

Example: This output shows that 11.19.140.0/24 matches criteria for the voice topology (community value is target:40:40), thus causing the route to be replicated in the :voice.inet.0 RIB (in addition to inet.0).

```
rtr-PE1> show route 11.19.140.0/24 terse
```

```

inet.0: 48 destinations, 49 routes (47 active, 0 holddown, 1 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

```

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.19.140.0/24	B	170	100		>11.19.1.2	102 I

```

:voice.inet.0: 22 destinations, 23 routes (21 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

```

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.19.140.0/24	B	170	100		>so-0/0/2.0	102 I

Example: IBGP routes in :voice.inet.0 use :voice.inet.0 to look up protocol next-hop address information. This protocol next hop has a forwarding next-hop IP address that forwards traffic for this IBGP route. Similarly, IBGP routes in :video.inet.0 look up protocol next hops in :video.inet.0. Note that if there is no route for the protocol next hop in the relevant RIB, the IBGP route state is hidden and unusable.

The protocol next hop of voice route 11.19.140.0/24 from the previous example is 10.255.164.203. This output shows this route in all of the topology RIBs. An important distinction is that voice and video each have a different forwarding next hop.

```

.....
rtr-PE1> show route 10.255.164.203 terse

inet.0: 48 destinations, 49 routes (47 active, 0 holddown, 1 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O 10      29             >11.19.1.2

:voice.inet.0: 22 destinations, 23 routes (21 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O 10      30             >so-0/0/2.0

:video.inet.0: 23 destinations, 23 routes (22 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination      P Prf  Metric 1   Metric 2   Next hop      AS path
* 10.255.164.203/32 O 10      29             >11.19.1.2
.....

```

As you can see, the forwarding next hop of 10.255.164.203 is different for voice and video. When route lookups use the RIB `:voice.inet.0`, the forwarding next-hop interface `so-0/0/2` is found. When route lookup takes place on the same protocol next-hop route 10.255.164.203 using a different RIB, `:video.inet.0`, the forwarding next hop is 11.19.1.2. The two traffic types can now be directed out different interfaces.

Example: Similar to voice routes, this PE1 output shows video BGP route 11.19.142.0/24 in `inet.0`. The community `target:50:50` designates the route as belonging to the video topology. The protocol next hop is 10.255.164.203, which is the same protocol next hop of the voice BGP route 11.19.140.0/24 previously shown.

```

.....
rtr-PE1> show route 11.19.142.0/24 extensive table inet.0

inet.0: 48 destinations, 49 routes (47 active, 0 holddown, 1 hidden)
11.19.142.0/24 (1 entry, 1 announced)
TSI:
KRT in-kernel 11.19.142.0/24 -> {indirect(262143)}
Page 0 idx 0 Type 1 val 8ab6720
  Nexthop: Self
  AS path: [100] 102 I
  Communities: target:50:50
Path 11.19.142.0 from 10.255.164.203 Vector len 4. Val: 0
  *BGP Preference: 170/-101
  Next hop type: Indirect
  Next-hop reference count: 9
  Source: 10.255.164.203
  Next hop type: Router, Next hop index: 560
  Next hop: 11.19.1.2 via ge-1/2/0.0, selected
Protocol next hop: 10.255.164.203
  Indirect next hop: 8b16000 262143
  State: <Active Int Ext>
  Local AS: 100 Peer AS: 100
  Age: 19:29 Metric2: 29
  Task: BGP_100.10.255.164.203+49384
  Announcement bits (3): 0-KRT 4-BGP RT Background 5-Resolve tree 3
  AS path: 102 I
Communities: target:50:50
  Localpref: 100
  Router ID: 10.255.164.203
.....

```

```

Secondary Tables: :video.inet.0
Indirect next hops: 1
    Protocol next hop: 10.255.164.203 Metric: 29
    Indirect next hop: 8b16000 262143
    Indirect path forwarding next hops: 1
        Next hop type: Router
        Next hop: 11.19.1.2 via ge-1/2/0.0
10.255.164.203/32 Originating RIB: inet.0
Metric: 29 Node path count: 1
Forwarding nexthops: 1
Nexthop: 11.19.1.2 via ge-1/2/0.0

```

Example: The previous example output shows route 11.19.142.0/24 matches criteria for the video topology (community value is target:50:50). This match causes the route, as shown below, to be replicated in the :video.inet.0 RIB (in addition to inet.0).

The forwarding next hop is derived from the protocol next hop 10.255.164.203, residing in :video.inet.0. Thus, the forwarding next hop of 11.19.142.0/24 is different from the forwarding next hop of voice route 11.19.140.0/24 (even though the protocol next hops are the same).

```
rtr-PE1> show route 11.19.142.0/24 terse
```

```
inet.0: 48 destinations, 49 routes (47 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both
```

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.19.142.0/24	B	170	100		>11.19.1.2	102 I

```
:video.inet.0: 21 destinations, 21 routes (20 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both
```

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.19.142.0/24	B	170	100		>11.19.1.2	102 I

Configuring the Forwarding Plane

The forwarding plane uses firewall filters to indicate which topology FIB traffic should use. In this case, you must configure a firewall filter on all interfaces related to routing topologies. In general, all MT-OSPF interfaces in the core where topologies are configured have input firewall filters. In addition, the ingress interfaces, where traffic from a CE enters a PE towards the core, have firewall filters configured.

Example: This PE1 configuration shows a firewall filter applied to the ingress interface (connected to the CE) and to the two core-facing interfaces (connected to P1 and P3).

```

interfaces {
    so-0/0/2 {
        description "PE1-P1";
        unit 0 {
            family inet {
                filter {
                    input ef_path;
                }
                address 11.19.6.1/24;
            }
        }
    }
    fe-0/2/3 {
        description "PE1-CE1";
        unit 0 {

```

```

        family inet {
            filter {
                input ef_path;
            }
            address 11.19.30.1/24;
        }
    }
}
ge-1/2/0 {
    description "PE1-P3";
    unit 0 {
        family inet {
            filter {
                input ef_path;
            }
            address 11.19.1.1/24;
        }
    }
}
}

```

Example: These firewall configurations show source addresses and DSCPs used to sort voice, video, or default traffic. The firewall filter uses most any filter that is recognizable. DSCPs are practical because you can set them at or near a CE router and because the information is intact across the network. For instance, here `class-of-service` is configured for expedited traffic. DSCPs are also practical when the same IP address is used for different applications.

```

firewall {
    family inet {
        filter ef_path {
            term ef {
                from {
                    forwarding-class expedited-forwarding; <- Filter on DSCP
                }
                then topology voice;
            }
            term video {
                from {
                    source-address {
                        11.19.132.0/24;
                        11.19.133.0/24;
                        11.19.142.0/24;
                        11.19.144.0/24;
                    }
                }
                then Topology video;
            }
            term default {
                then accept;
            }
        }
    }
}
class-of-service {
    interfaces {
        so-0/0/2 {
            unit 0 {
                classifiers {
                    inet-precedence default;
                }
            }
        }
    }
}

```


Example: This output shows a `traceroute` from CE1 to CE2 for video traffic where the firewall filter is based on the destination address. The routes are resolved over `:video.inet.0`. As shown in Figure 2, this `traceroute` follows the video path CE1-PE1-P3-P4-PE2-CE2.

```

rtr-CE1> traceroute 11.19.142.1 source 11.19.132.1
traceroute to 11.19.142.1 (11.19.142.1) from 11.19.132.1, 30 hops max, 40 byte packets
 1 11.19.30.1 (11.19.30.1) 0.830 ms 0.655 ms 0.644 ms
 2 11.19.1.2 (11.19.1.2) 9.976 ms 0.592 ms 0.545 ms <- This hop is via P3
 3 11.19.17.2 (11.19.17.2) 0.740 ms 0.773 ms 0.670 ms
 4 11.19.27.1 (11.19.27.1) 0.583 ms 0.600 ms 0.579 ms
 5 11.19.142.1 (11.19.142.1) 0.964 ms 0.905 ms 0.851 ms

```

Example: This output shows a `traceroute` from CE1 to CE2 for video where DSCPs are set. Note that the DSCP bits are directing PE1 to look in topology table `:voice.inet.0`. Since there is no entry in the voice RIB for the video routes, traffic is dropped.

```

rtr-CE1> traceroute 11.19.142.1 source 11.19.132.1 tos 160
traceroute to 11.19.142.1 (11.19.142.1) from 11.19.132.1, 30 hops max, 40 byte packets
 1 11.19.30.1 (11.19.30.1) 0.800 ms !N 0.666 ms !N 0.600 ms !N <- This drop is expected

```

Case Study: Choosing a Topology Path Based on a Multicast Source

In this case study, assume you want to provide redundancy for multicast traffic over separate network paths. That is, you intend for the two multicast sources to send the same multicast stream, yet for redundancy purposes in the case of link failure, you require that the two streams use disjoint paths. Note there is no standard defined at this time for using MTR extensions to PIM.

Assume that each source providing redundant multicast streams, S1 and S2, have different IP subnet addresses. Each source sends multicast traffic using different groups: G1 and G2. Further, assume that S1 and S2 are attached to the same router and use BGP to announce routes to the PE router.

The Junos OS solution provides a mechanism whereby multicast traffic traverses user-specified topology paths based on the sender's source address. MTR is used for OSPF, BGP, and route resolution over the specified topology RIBs. Although basic PIM is used, it has no need to know about MTR since OSPF and BGP independently populate the RIB table used by PIM. Firewall filters are not required because the multicast forwarding plane uses the multicast tree once it has been built. For a complete set of configurations for PE1, PE2, P1, and CE1, see *Appendix B*.

Figure 3 shows a simplified diagram of routing topology paths, where the dashed lines are associated with multicast group "A" (topology red) and the dotted lines are associated with multicast group "B" (topology blue). Two copies of the same stream enter PE1 and then traverse separate paths over the IBGP core.

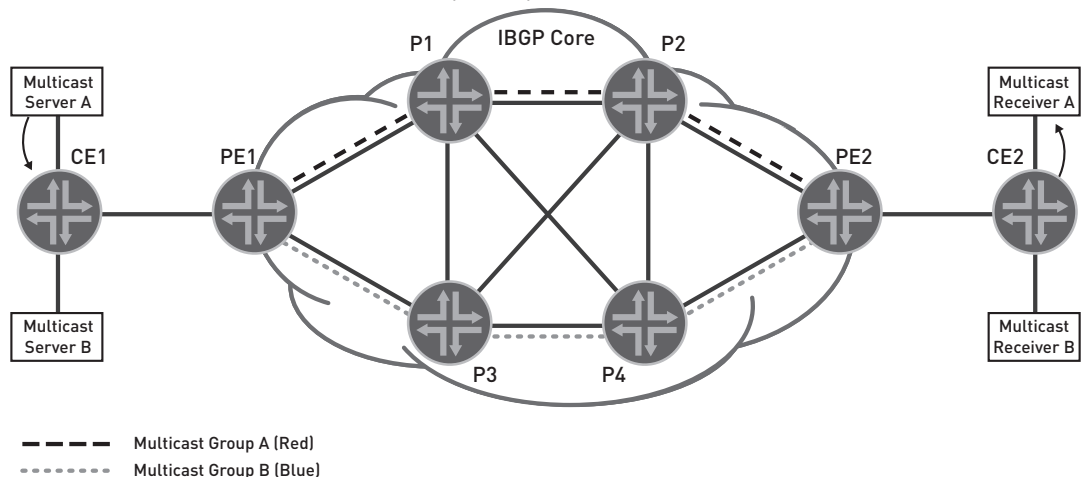


Figure 3: IBGP core links configured to prefer specified routing topologies

This solution leverages Junos OS features that allow particular RIBs to perform route resolution using specified RIBs. The configuration includes a combination of the following.

- BGP communities
- Separate IBGP next hops belonging to user-specified OSPF routing topologies
- Route resolution over user-specified topology RIBs
- A separate routing table (`inet.2`) for multicast protocols

Commonly, networks use a separate routing table for multicast, which is often referred to as a *multicast RIB (MRIB)*. While an MRIB is effectively `inet.2`, the term *MRIB* is referred to throughout this case study.

Routing topologies are grouped based on BGP communities. Each group represents a set of IP addresses associated with multicast servers and receivers. Primarily the group must be related to the set of servers because the multicast receivers initiate tree creation towards these servers. Multicast traffic directed downstream toward receivers uses the previously created PIM tree, and therefore the forwarding plane does not need to know about routing topologies.

PIM uses the MRIB for route lookups on multicast source addresses. Note that these IP addresses used for tree creation are IP unicast addresses. The CE routers, nearest to the multicast servers, announce the multicast source IP addresses to the PE routers using EBGP. They are announced with both family `inet` unicast and family `inet` multicast, thus causing the BGP route to be added to the default RIB `inet.0` and to the MRIB `inet.2`. Both are injected by the PE router into IBGP.

Each BGP route injected into IBGP has a specific protocol next hop. Junos OS provides the flexibility to set the protocol next hop when exporting the route into IBGP (for instance, `next-hop self` could be set via an export policy configuration). You can also set the protocol next hop to a route associated with a specified topology RIB. Keeping in mind that an EBGP route can have a community associated with a routing topology, you can conveniently configure a policy to use this community to designate which protocol next hop should be set when exporting the IBGP route into `inet.2`. As such, a specific protocol next-hop IP address is required per topology on each router injecting IBGP routes. You can configure multiple secondary loopback IP addresses on a router to be used as protocol next-hop addresses.

Example: This configuration shows nonprimary IP addresses `1.1.1.30/32` and `2.2.2.30/32` configured on loopback interface `lo0` for use in the red and blue topologies, respectively.

```
.....
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 10.255.165.93/32 { <- Primary default loopback
                    primary;
                }
                address 1.1.1.30/32;      <- Loopback for the red topology
                address 2.2.2.30/32;      <- Loopback for the blue topology
            }
        }
    }
}
.....
```

A group of BGP routes associated with a routing topology use the same unique protocol next hop. For instance, if you configure a PE router to handle two routing topologies, then you would also configure two unique nonprimary addresses under loopback interface `lo0`.

Next, associate each nonprimary loopback IP address with a topology for inclusion in the associated topology RIB. Configure the loopback IP address and topology under an OSPF interface statement. You must specifically disable all other topologies known to OSPF for two reasons. First, the loopback address specific to a topology must reside in only one topology RIB. Second, once the topology is added to the top stanza of OSPF, the topology defaults to being enabled on all subsequent interfaces under OSPF.

Example: This PE1 configuration places the loopback address 1.1.1.30/32 into the OSPF database as a stub route under this router's OSPF Router-LSA. It will belong to the red and default topologies, but not to the blue topology. The loopback address 1.1.1.30/32 will be installed in remote core routers' topology RIBs `inet.0` and `:red.inet.0`, (but not in `:blue.inet.0`). Use a similar configuration for the blue loopback address 2.2.2.30/32.

```

.....
protocols {
  ospf {
    topology red topology-id 126;
    topology blue topology-id 52;
    area 0.0.0.0 {
      interface 1.1.1.30 {
        topology red;
        topology blue disable;
      }
      interface 2.2.2.30 {
        topology blue;
        topology red disable;
      }
    }
  }
}
.....

```

Example: This PE1 output shows the Router-LSA originated by this router (the PE1 `router-id` is 10.255.165.93). This output also shows the secondary loopback addresses as stubs, including to which topology they belong.

```

.....
rtr-pe1> show ospf database router advertising-router 10.255.165.93 extensive

  OSPF link state database, Area 0.0.0.0
Type      ID                Adv Rtr           Seq             Age  Opt  Cksum  Len
Router *10.255.165.93  10.255.165.93    0x8000001f      1796  0x22  0x5568  112
bits 0x0, link count 5
id 11.19.1.2, data 11.19.1.1, Type Transit (2)
  Topology count: 2, Default metric: 10
  Topology blue (ID 52) -> Metric: 1
  Topology red (ID 126) -> Metric: 10
id 11.19.6.2, data 11.19.6.1, Type Transit (2)
  Topology count: 2, Default metric: 10
  Topology blue (ID 52) -> Metric: 10
  Topology red (ID 126) -> Metric: 1
id 10.255.165.93, data 255.255.255.255, Type Stub (3)
  Topology count: 1, Default metric: 10
  Topology red (ID 126) -> Metric: 10
id 1.1.1.30, data 255.255.255.255, Type Stub (3) <- 1.1.1.30/32 is a stub in default and red topologies
  Topology count: 1, Default metric: 10
  Topology red (ID 126) -> Metric: 10
id 2.2.2.30, data 255.255.255.255, Type Stub (3) <- 2.2.2.30/32 is a stub in default and blue topologies
  Topology count: 1, Default metric: 10
  Topology blue (ID 52) -> Metric: 10
Gen timer 00:20:03
Aging timer 00:30:03
Installed 00:29:56 ago, expires in 00:30:04, sent 00:29:54 ago
Last changed 22:59:56 ago, Change count: 3, Ours
.....

```

The next configuration on the PE router is to notify PIM which RIB to use.

Example: This basic configuration creates an MRIB and indicates to PIM that it should use the MRIB route table `inet.2`. Note that these configurations are not related to routing topologies.

```

.....
routing-options {
  rib-groups {
    MRIB {
      import-rib inet.2;
    }
  }
}
protocols {
  pim {
    rib-group inet MRIB;
  }
}
.....

```

You can configure the router to perform route resolution on protocol next hops using specified RIBs. The protocol next hop is then used to determine the forwarding next-hop interface out which to forward PIM joins.

Example: This PE1 configuration directs `inet.2` route resolution to use topology RIBs `:red.inet.0` and `:blue.inet.0` for protocol next-hop IP address lookups.

```

.....
routing-options {
  resolution {
    rib inet.2 {
      resolution-ribs [ :red.inet.0 :blue.inet.0 ];
    }
  }
}
.....

```

You can specify up to two RIBs in the resolution configuration. Note that a key element to this solution is that the protocol next-hop address resides in only one topology RIB. That is, the protocol next hop belongs to a remote PE secondary loopback address and is injected into only one topology RIB. The route resolution scheme first checks RIB `:red.inet.0` for the protocol next-hop address. If the address is found, it uses this entry. If it is not found, the resolution scheme then checks RIB `:blue.inet.0`. Hence, only one topology RIB is used for each protocol next-hop address.

Links can support all routing topologies to provide backup should a primary multicast path fail. You can configure specific OSPF link metrics on topologies to identify paths and build trees to different servers. When a multicast tree gets built via PIM `join` messages directed towards the source, it follows the most preferred path. A multicast tree to a different multicast source (in a different routing topology) can create another tree along a different path.

Figure 4 shows an example of two trees using different paths over different topologies. It shows Server A using the multicast tree with the dashed line as its path and Server B using the multicast tree with the dotted line as its path.

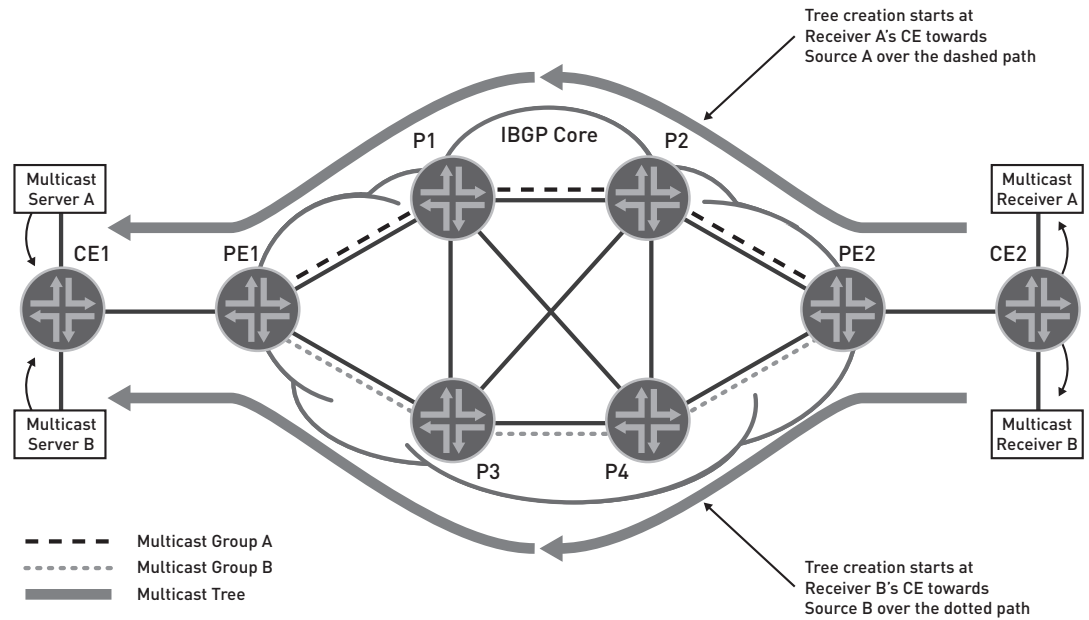


Figure 4: Two multicast trees traversing two different paths via routing topologies

Example: This configuration shows the ingress PE1 router sets the BGP route's protocol next-hop address when exporting the route into IBGP. It shows how BGP uses an export policy to set the next hop when injecting the EBGp routes into IBGP.

```

protocols {
  bgp {
    group ibgp {
      type internal;
      local-address 10.255.165.93;
      family inet {
        unicast;
        multicast;
      }
      export nexthop_three_selves;    <- Indicates how the next hops get set
      neighbor 10.255.165.97;
      ...
    }
  }
}

```

Example: This configuration is the export policy used from the previous example where there are three possibilities of next hops being set.

- 1.1.1.30 — associated with the red topology
- 2.2.2.30 — associated with the blue topology
- Default next-hop self — the primary PE1 loopback address 10.255.165.93.

```

policy-options {
  policy-statement nexthop_three_selves {
    term red {
      from {
        protocol bgp;
        community red;
      }
      then {

```

```

        next-hop 1.1.1.30;
        accept;
    }
}
term blue {
    from {
        protocol bgp;
        community blue;
    }
    then {
        next-hop 2.2.2.30;
        accept;
    }
}
term default {
    then {
        next-hop self;
    }
}
}
}
}

```

The CE router advertises routes via EBGP to the PE router. These routes are advertised as BGP family `inet` multicast routes with communities set for two different groups. Policies identify the two groups of BGP routes.

Example: This CE1 configuration shows how to advertise these routes via EBGP to the PE.

```

protocols {
    bgp {
        group ebgp {
            type external;
            local-address 11.19.30.2;
            family inet {
                unicast;
                multicast;
            }
            export set_community;
            peer-as 100;
            neighbor 11.19.30.1;
        }
    }
}
policy-options {
    policy-statement set_community {
        term a {
            from {
                route-filter 11.19.130.0/24 exact;
                route-filter 11.19.131.0/24 exact;
            }
            then {
                community add red;
                accept;
            }
        }
        term b {
            from {
                route-filter 11.19.132.0/24 exact;
                route-filter 11.19.133.0/24 exact;
            }
            then {

```

```

        community add blue;
        accept;
    }
}
term default {
    then accept;
}
}
community blue members target:50:50;
community red members target:40:40;
}

```

Example: This output shows two IBGP routes in `inet.2` as seen from PE2. These routes were originally injected into IBGP by PE1, where their next hops were set based on the topology to which they belonged. The BGP community value determined the topology association.

The first route, `11.19.130/24`, belongs to the red topology since it has a community value of `target:40:40`. Note that the protocol next hop is `1.1.1.30`, and the forwarding next hop is via `ge-0/1/3.0`.

```

rtr-PE2> show route 11.19.130.0/24 table inet.2 extensive

inet.2: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
11.19.130.0/24 (1 entry, 1 announced)
TSI:
Page 0 idx 0 Type 1 val 8e88f48
  Nexthop: Self
  AS path: [100] 101 I
  Communities: target:40:40
Path 11.19.130.0 from 10.255.165.93 Vector len 4. Val: 0
  *BGP   Preference: 170/-101
        Next hop type: Indirect
        Next-hop reference count: 2
        Source: 10.255.165.93
        Next hop type: Router, Next hop index: 628
        Next hop: 11.19.25.2 via ge-0/1/3.0, selected <- Forwarding hop
        Protocol next hop: 1.1.1.30
        Indirect next hop: 8cb0640 -
        State: <Active Int Ext>
        Local AS: 100 Peer AS: 100
        Age: 47:26 Metric2: 3
        Task: BGP_100.10.255.165.93+179
        Announcement bits (1): 0-BGP RT Background
        AS path: 101 I
        Communities: target:40:40                                <- Indicates red topology
        Accepted
        Localpref: 100
        Router ID: 10.255.165.93
        Indirect next hops: 1
            Protocol next hop: 1.1.1.30 Metric: 3 <- Protocol next hop
            Indirect next hop: 8cb0640 -
            Indirect path forwarding next hops: 1
                Next hop type: Router
                Next hop: 11.19.25.2 via ge-0/1/3.0
1.1.1.30/32 Originating RIB: :red.inet.0
  Metric: 3 Node path count: 1
  Forwarding nexthops: 1
  Nexthop: 11.19.25.2 via ge-0/1/3.0

```

Example: The second route, 11.19.132/24, belongs to the blue topology since it has a community value of target:50:50. The protocol next hop is 2.2.2.30, and the forwarding next hop is via ge-0/0/3.0.

These two BGP routes show how it is possible to use different protocol next hops when originating the route from the same IBGP speaker, PE1 in this case. It shows how by doing this, the forwarding next hops are different, causing traffic to take a different path when it exits PE2 towards PE1.

```

.....
inet.2: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
11.19.132.0/24 (1 entry, 1 announced)
TSI:
Page 0 idx 0 Type 1 val 8e88f78
  Nexthop: Self
  AS path: [100] 101 I
  Communities: target:50:50
Path 11.19.132.0 from 10.255.165.93 Vector len 4. Val: 0
  *BGP Preference: 170/-101
    Next hop type: Indirect
    Next-hop reference count: 1
    Source: 10.255.165.93
    Next hop type: Router, Next hop index: 605
    Next hop: 11.19.27.2 via ge-0/0/3.0, selected <- Forwarding hop
    Protocol next hop: 2.2.2.30
    Indirect next hop: 8cb06e0 -
    State: <Active Int Ext>
    Local AS: 100 Peer AS: 100
    Age: 47:28 Metric2: 3
    Task: BGP_100.10.255.165.93+179
    Announcement bits (1): 0-BGP RT Background
    AS path: 101 I
    Communities: target:50:50 <- Indicates blue topology
    Accepted
    Localpref: 100
    Router ID: 10.255.165.93
    Indirect next hops: 1
      Protocol next hop: 2.2.2.30 Metric: 3 <- Protocol next hop
      Indirect next hop: 8cb06e0 -
      Indirect path forwarding next hops: 1
        Next hop type: Router
        Next hop: 11.19.27.2 via ge-0/0/3.0
      2.2.2.30/32 Originating RIB: :blue.inet.0
      Metric: 3 Node path count: 1
      Forwarding nexthops: 1
        Nexthop: 11.19.27.2 via ge-0/0/3.0
.....

```

Conclusion

The MTR solution in Junos OS provides a powerful mechanism for traffic engineering IP unicast and multicast traffic and as a potential augmentation to MPLS. You can use MTR in conjunction with direct routes, static routes, OSPF, and BGP. Junos OS also supports disjoint paths in PIM so that you can have true redundancy without any changes to the existing PIM.

Acronyms

BGP	Border Gateway Protocol
CE	customer edge
CLI	command-line interface
DSCP	DiffServe code point
EBGP	external BGP
FIB	forwarding information base
G	group
IBGP	internal BGP
IGP	interior gateway protocol
LSA	link-state advertisement
MPLS	Multiprotocol Label Switching
MRIB	multicast RIB
MTR	multi-topology routing
OSPF	Open Shortest Path First
P	provider
PE	provider edge
PIM	Protocol Independent Multicast
RIB	routing information base
ToS	type of service

Appendix A: Configurations for the Case Study on Choosing a Topology Path Based on an Application

PE1 Router Configurations

```

.....
interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 10.255.165.93/32 {
          primary;
        }
      }
    }
  }
  so-0/0/2 {
    unit 0 {
      family inet {
        filter {
          input ef_path;
        }
        address 11.19.6.1/24;
      }
      family mpls;
    }
  }
  fe-0/2/3 {
    unit 0 {
      family inet {
        filter {
          input ef_path;
        }
        address 11.19.30.1/24;
      }
    }
  }
}

```

```
        family mpls;
    }
}
ge-1/2/0 {
    unit 0 {
        family inet {
            filter {
                input ef_path;
            }
            address 11.19.1.1/24;
        }
        family mpls;
    }
}
}
routing-options {
    autonomous-system 100;
    topologies {
        family inet {
            topology voice;
            topology video;
        }
    }
}
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.255.165.93;
            family inet {
                unicast {
                    topology voice {
                        community target:40:40;
                    }
                    topology video {
                        community target:50:50;
                    }
                }
            }
        }
        export nhs;
        neighbor 10.255.165.111;
        neighbor 10.255.164.203;
        neighbor 10.255.165.113;
        neighbor 10.255.165.95;
        neighbor 10.255.165.99;
    }
    group ebgp {
        type external;
        local-address 11.19.30.1;
        family inet {
            unicast {
                topology voice {
                    community target:40:40;
                }
                topology video {
                    community target:50:50;
                }
            }
        }
    }
    peer-as 101;
    neighbor 11.19.30.2;
}
```

```

    }
  }
  ospf {
    topology voice topology-id 126;
    topology video topology-id 52;
    area 0.0.0.0 {
      interface ge-1/2/0.0 {
        metric 10;
        topology video;
        topology voice disable;
      }
      interface so-0/0/2.0 {
        metric 10;
        topology voice;
        topology video disable;
      }
      interface lo0.0 {
        passive;
      }
    }
  }
}
policy-options {
  policy-statement nhs {
    then {
      next-hop self;
    }
  }
}
firewall {
  family inet {
    filter ef_path {
      term ef {
        from {
          forwarding-class expedited-forwarding;
        }
        then topology voice;
      }
      term video {
        from {
          source-address {
            11.19.132.0/24;
            11.19.133.0/24;
            11.19.142.0/24;
            11.19.144.0/24;
          }
        }
        then topology video;
      }
      term default {
        then accept;
      }
    }
  }
}
class-of-service {
  interfaces {
    so-0/0/2 {
      unit 0 {
        classifiers {
          inet-precedence default;
        }
      }
    }
  }
}

```



```

        }
        address 11.19.25.1/24;
    }
    family mpls;
}
}
}
routing-options {
    autonomous-system 100;
    topologies {
        family inet {
            topology voice;
            topology video;
        }
    }
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.255.164.203;
            family inet {
                unicast {
                    topology voice {
                        community target:40:40;
                    }
                    topology video {
                        community target:50:50;
                    }
                }
            }
            export nhs;
            neighbor 10.255.165.93;
            neighbor 10.255.165.111;
            neighbor 10.255.165.113;
            neighbor 10.255.165.95;
            neighbor 10.255.165.99;
        }
        group ebgp {
            type external;
            local-address 11.19.40.1;
            family inet {
                unicast {
                    topology voice {
                        community target:40:40;
                    }
                    topology video {
                        community target:50:50;
                    }
                }
            }
            peer-as 102;
            neighbor 11.19.40.2;
        }
    }
    ospf {
        topology voice topology-id 126;
        topology video topology-id 52;
        area 0.0.0.0 {
            interface ge-0/1/4.0 {
                description "PE2 <-> P2";
            }
        }
    }
}

```

```
        metric 10;
        topology voice;
        topology video metric 200;
    }
    interface ge-0/0/3.0 {
        description "PE2 <-> P4";
        metric 10;
        topology voice disable;
        topology video;
    }
    interface lo0.0 {
        passive;
    }
}
}
}
policy-options {
    policy-statement nhs {
        then {
            next-hop self;
        }
    }
}
}
firewall {
    family inet {
        filter ef_path {
            term ef {
                from {
                    forwarding-class expedited-forwarding;
                }
                then topology voice;
            }
            term video {
                from {
                    source-address {
                        11.19.132.0/24;
                        11.19.133.0/24;
                        11.19.142.0/24;
                        11.19.144.0/24;
                    }
                }
                then topology video;
            }
            term default {
                then accept;
            }
        }
    }
}
}
}
class-of-service {
    interfaces {
        ge-0/0/3 {
            unit 0 {
                classifiers {
                    inet-precedence default;
                }
            }
        }
        ge-0/0/4 {
            unit 0 {
                classifiers {
```



```
ge-4/0/7 {
  unit 0 {
    family inet {
      filter {
        input ef_path;
      }
      address 11.19.57.1/24;
    }
    family mpls;
  }
}
}
routing-options {
  autonomous-system 100;
  topologies {
    family inet {
      topology voice;
      topology video;
    }
  }
}
protocols {
  bgp {
    group ibgp {
      type internal;
      local-address 10.255.165.113;
      family inet {
        unicast {
          topology voice {
            community target:40:40;
          }
          topology video {
            community target:50:50;
          }
        }
      }
      neighbor 10.255.165.93;
      neighbor 10.255.165.111;
      neighbor 10.255.164.203;
      neighbor 10.255.165.95;
      neighbor 10.255.165.99;
    }
  }
}
ospf {
  topology voice topology-id 126;
  topology video topology-id 52;
  area 0.0.0.0 {
    interface ge-4/0/6.0 {
      description "P2 <-> P1";
      metric 10;
      topology voice;
      topology video disable;
    }
    interface ge-4/0/2.0 {
      description "P2 <-> P3";
      metric 10;
      topology voice disable;
      topology video metric 20;
    }
    interface ge-4/0/7.0 {
      description "P2 <-> P4";
    }
  }
}
```



```
}
policy-options {
  policy-statement inject_directs {
    term a {
      from {
        protocol direct;
        interface fe-1/1/3.0;
      }
      then {
        next policy;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement set_community {
    term a {
      from {
        route-filter 11.19.130.0/24 exact;
        route-filter 11.19.131.0/24 exact;
      }
      then {
        community add voice;
        accept;
      }
    }
    term b {
      from {
        route-filter 11.19.132.0/24 exact;
        route-filter 11.19.133.0/24 exact;
      }
      then {
        community add video;
        accept;
      }
    }
    term default {
      then accept;
    }
  }
  community voice members target:40:40;
  community video members target:50:50;
}
.....
```

Appendix B: Configurations for the Case Study on Choosing a Topology Path Based on a Multicast Source

PE1 Router Configurations

```
interfaces {
  /*
  * Secondary addresses 1.1.1.30 and 2.2.2.30 are configured.
  */
  lo0 {
    unit 0 {
      family inet {
        address 127.0.0.1/32;
        address 10.255.165.93/32 {
          primary;
        }
        address 1.1.1.30/32;
        address 2.2.2.30/32;
      }
    }
  }
  so-0/0/2 {
    unit 0 {
      family inet {
        address 11.19.30.1/24;
      }
      family mpls;
    }
  }
  fe-0/2/3 {
    unit 0 {
      family inet {
        address 11.19.1.1/24;
      }
      family mpls;
    }
  }
  ge-1/2/1 {
    unit 0 {
      family inet {
        address 11.19.6.1/24;
      }
      family mpls;
    }
  }
}
routing-options {
  /*
  * Set up a separate route group, MRIB, for multicast lookups. It will
  * be populated with routes from inet.2. inet.2 is populated by BGP
  * BGP routes with type "multicast".
  */
  rib-groups {
    MRIB {
      import-rib inet.2;
    }
  }
  autonomous-system 100;
  /*
```

```

* During route resolution of inet.2, use routing topology "red" to
* lookup the protocol next hop's forwarding next hop. If the protocol
* next hop is not present, then check routing topology "blue".
*/
resolution {
    rib inet.2 {
        resolution-ribs [ :red.inet.0 :blue.inet.0 ];
    }
}
/*
* Configure routing Topologies red and blue
*/
topologies {
    family inet {
        topology red;
        topology blue;
    }
}
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.255.165.93;
            /*
            * Both unicast and multicast BGP prefixes are exchanged.
            */
            family inet {
                unicast;
                multicast;
            }
            /*
            * Set the protocol next hop when exporting EBGp into IBGP.
            */
            export [ nhs_test nhs_inet0_self ];
            neighbor 10.255.165.97;
            neighbor 10.255.164.203;
            neighbor 10.255.165.111;
            neighbor 10.255.165.113;
            neighbor 10.255.165.99;
        }
        group ebgp {
            type external;
            local-address 11.19.30.1;
            family inet {
                unicast;
                multicast;
            }
            peer-as 101;
            neighbor 11.19.30.2;
        }
    }
}
ospf {
    topology red topology-id 126;
    topology blue topology-id 52;
    area 0.0.0.0 {
        interface fe-0/2/3.0 {
            metric 10;
            topology red;
            topology blue metric 1;
        }
    }
}

```

```
interface ge-1/2/1.0 {
    metric 10;
    topology red metric 1;
    topology blue;
}
interface lo0.0 {
    passive;
}
/*
 * This places the secondary loopback address, 1.1.1.30, into
 * as a stub route under this routers Router-LSA. It will
 * belong to topology red and default, but not blue. It will
 * then land in remote routers in inet.0 and :red.inet.0,
 * but not in :blue.inet.0.
 */
interface 1.1.1.30 {
    topology red;
    topology blue disable;
}
/*
 * This places the secondary loopback address, 2.2.2.30, into
 * as a stub route under this routers Router-LSA. It will
 * belong to topology blue and default, but not red. It will
 * then land in remote routers in inet.0 and :blue.inet.0, but
 * not in :red.inet.0
 */
interface 2.2.2.30 {
    topology blue;
    topology red disable;
}
}
}

pim {
    /*
     * This tells PIM to use the routes in inet.2 (which populate the MRIB).
     */
    rib-group inet MRIB;
    interface fe-0/2/3.0 {
        mode sparse;
    }
    interface so-0/0/2.0 {
        mode sparse;
    }
    interface ge-1/2/1.0 {
        mode sparse;
    }
}

policy-options {
    /*
     * This policy ensures that all BGP prefixes in inet.0 always
     * use the primary loopback address of this router as protocol
     * next hop.
     */
    policy-statement nhs_inet0_self {
        term a {
            from {
                protocol bgp;
                rib inet.0;
            }
        }
    }
}
```

```
        then {
            next-hop self;
        }
    }
}
/*
 * This policy sets the protocol next-hop based on the community
 * in the BGP update.
 */
policy-statement nhs_test {
    term a {
        from {
            protocol bgp;
            community red;
        }
        then {
            next-hop 1.1.1.30;
            next policy;
            accept;
        }
    }
    term b {
        from {
            protocol bgp;
            community blue;
        }
        then {
            next-hop 2.2.2.30;
            next policy;
            accept;
        }
    }
    term c {
        then {
            next-hop self;
        }
    }
}
community blue members target:50:50;
community red members target:40:40;
}
```

PE2 Router Configurations

```
interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 10.255.164.203/32 {
          primary;
        }
        address 1.1.1.40/32;
        address 2.2.2.40/32;
      }
    }
  }
  ge-0/0/3 {
    unit 0 {
      family inet {
        address 11.19.27.1/24;
      }
      family mpls;
    }
  }
  ge-0/0/4 {
    unit 0 {
      family inet {
        address 11.19.40.1/24;
      }
      family mpls;
    }
  }
  ge-0/1/3 {
    unit 0 {
      family inet {
        address 11.19.25.1/24;
      }
      family mpls;
    }
  }
}
routing-options {
  rib-groups {
    MRIB {
      import-rib inet.2;
    }
  }
  autonomous-system 100;
  resolution {
    rib inet.2 {
      resolution-ribs [ :red.inet.0 :blue.inet.0 ];
    }
  }
  topologies {
    family inet {
      topology red;
      topology blue;
    }
  }
}
protocols {
  bgp {
```

```
group ibgp {
    type internal;
    local-address 10.255.164.203;
    family inet {
        unicast;
        multicast;
    }
    export [ nhs_test nhs_inet0_self ];
    neighbor 10.255.165.93;
    neighbor 10.255.165.97;
    neighbor 10.255.165.111;
    neighbor 10.255.165.113;
    neighbor 10.255.165.99;
}
group ebgp {
    type external;
    local-address 11.19.40.1;
    family inet {
        unicast;
        multicast;
    }
    peer-as 102;
    neighbor 11.19.40.2;
}
}
ospf {
    topology red topology-id 126;
    topology blue topology-id 52;
    area 0.0.0.0 {
        interface ge-0/1/3.0 {
            metric 10;
            topology red metric 1;
            topology blue;
        }
        interface ge-0/0/3.0 {
            metric 10;
            topology red;
            topology blue metric 1;
        }
        interface lo0.0 {
            passive;
        }
        interface 1.1.1.40 {
            topology red;
            topology blue disable;
        }
        interface 2.2.2.40 {
            topology blue;
            topology red disable;
        }
    }
}
}
pim {
    rib-group inet MRIB;
    interface ge-0/0/4.0 {
        mode sparse;
    }
    interface ge-0/1/3.0 {
        mode sparse;
    }
    interface ge-0/0/3.0 {
```

```
        mode sparse;
    }
}
}
policy-options {
    policy-statement nhs {
        then {
            next-hop self;
        }
    }
    policy-statement nhs_inet0_self {
        term a {
            from {
                protocol bgp;
                rib inet.0;
            }
            then {
                next-hop self;
            }
        }
    }
    policy-statement nhs_test {
        term a {
            from {
                protocol bgp;
                community red;
            }
            then {
                next-hop 1.1.1.40;
                next policy;
                accept;
            }
        }
        term b {
            from {
                protocol bgp;
                community blue;
            }
            then {
                next-hop 2.2.2.40;
                next policy;
                accept;
            }
        }
        term c {
            then {
                next-hop self;
            }
        }
    }
    community red members target:40:40;
    community blue members target:50:50;
}
.....
```

P1 Router Configurations

```
interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 127.0.0.1/32;
        address 10.255.165.97/32 {
          primary;
        }
      }
    }
  }
  tl-0/1/3 {
    unit 0 {
      family inet {
        address 11.19.17.1/24;
      }
      family mpls;
    }
  }
  fe-0/2/3 {
    unit 0 {
      family inet {
        address 11.19.1.2/24;
      }
      family mpls;
    }
  }
  ge-1/2/0 {
    unit 0 {
      family inet {
        address 11.19.15.1/24;
      }
      family mpls;
    }
  }
  ge-1/2/1 {
    unit 0 {
      family inet {
        address 11.19.16.1/24;
      }
      family mpls;
    }
  }
}
routing-options {
  rib-groups {
    MRIB {
      import-rib inet.2;
    }
  }
  autonomous-system 100;
  topologies {
    family inet {
      topology red;
      topology blue;
    }
  }
}
```

```
protocols {
  bgp {
    group ibgp {
      type internal;
      local-address 10.255.165.97;
      family inet {
        unicast;
        multicast;
      }
      neighbor 10.255.165.93;
      neighbor 10.255.164.203;
      neighbor 10.255.165.111;
      neighbor 10.255.165.113;
      neighbor 10.255.165.99;
    }
  }
  ospf {
    topology red topology-id 126;
    topology blue topology-id 52;
    area 0.0.0.0 {
      interface fe-0/2/3.0 {
        metric 10;
        topology red;
        topology blue metric 1;
      }
      interface ge-1/2/0.0;
      interface ge-1/2/1.0;
      interface t1-0/1/3.0 {
        metric 10;
        topology red;
        topology blue metric 1;
      }
      interface lo0.0 {
        passive;
      }
    }
  }
  pim {
    rib-group inet MRIB;
    interface fe-0/2/3.0 {
      mode sparse;
    }
    interface ge-1/2/0.0 {
      mode sparse;
    }
    interface ge-1/2/1.0 {
      mode sparse;
    }
    interface t1-0/1/3.0 {
      mode sparse;
    }
  }
}
```

CE1 Router Configurations

Note that this CE configuration uses direct routes and `rib-groups` to simulate a group of BGP routes with communities attached and announced as multicast routes.

```

.....
interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 10.255.165.95/32 {
          primary;
        }
      }
    }
  }
  so-0/0/2 {
    unit 0 {
      family inet {
        address 11.19.30.2/24;
      }
    }
  }
  fe-1/1/3 {
    fastether-options {
      loopback;
    }
    unit 0 {
      family inet {
        address 11.19.130.1/24;
        address 11.19.131.1/24;
        address 11.19.132.1/24;
      }
    }
  }
}
routing-options {
  interface-routes {
    rib-group inet if-rib;
  }
  static {
    route 11.19.0.0/16 next-hop 11.19.30.1;
  }
  rib-groups {
    inet.2 {
      import-rib inet.0;
    }
    if-rib {
      import-rib [ inet.0 inet.2 ];
      import-policy inject_directs;
    }
  }
  autonomous-system 101;
}
protocols {
  bgp {
    group ebgp {
      type external;
      local-address 11.19.30.2;
      family inet {
        unicast;
      }
    }
  }
}

```

```
        multicast;
    }
    export [ inject_directs set_community ];
    peer-as 100;
    neighbor 11.19.30.1;
}
pim {
    interface fe-1/1/3.0 {
        mode sparse;
    }
    interface so-0/0/2.0 {
        mode sparse;
    }
}
}
policy-options {
    policy-statement inject_directs {
        term a {
            from {
                protocol direct;
                interface fe-1/1/3.0;
            }
            then {
                next policy;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    policy-statement set_community {
        term a {
            from {
                route-filter 11.19.130.0/24 exact;
                route-filter 11.19.131.0/24 exact;
            }
            then {
                community add red;
                accept;
            }
        }
        term b {
            from {
                route-filter 11.19.132.0/24 exact;
                route-filter 11.19.133.0/24 exact;
            }
            then {
                community add blue;
                accept;
            }
        }
        term default {
            then accept;
        }
    }
    community red members target:40:40;
    community blue members target:50:50;
}
}
```

For More Information

Multi-Topology (MT) Routing in OSPF - RFC 4915. <http://tools.ietf.org/html/rfc4915>.

About Juniper Networks

Juniper Networks, Inc. is the leader in high-performance networking. Juniper offers a high-performance network infrastructure that creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network. This fuels high-performance businesses. Additional information can be found at www.juniper.net.

Corporate and Sales Headquarters

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER [888.586.4737]
or 408.745.2000
Fax: 408.745.2100
www.juniper.net

APAC Headquarters

Juniper Networks (Hong Kong)
26/F, Cityplaza One
1111 King's Road
Taikoo Shing, Hong Kong
Phone: 852.2332.3636
Fax: 852.2574.7803

EMEA Headquarters

Juniper Networks Ireland
Airsides Business Park
Swords, County Dublin, Ireland
Phone: 35.31.8903.600
EMEA Sales: 00800.4586.4737
Fax: 35.31.8903.601

To purchase Juniper Networks solutions, please contact your Juniper Networks representative at 1-866-298-6428 or authorized reseller.

Copyright 2010 Juniper Networks, Inc. All rights reserved. Juniper Networks, the Juniper Networks logo, Junos, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

2000308-002-EN Aug 2010

 Printed on recycled paper