JUNIPER NETWORKS

Junos® Learning Sphere

junosphere

vDAY ONE: INTRODUCTION TO BGP MULTICAST VPNs

10 VM - 4.5 hrs

Let's build a BGP Multicast VPN topology by using a half dozen key Junos configurations. Log on to Junosphere and learn how. Right now.

by Antonio Sánchez-Monge

This new *vDay One* book is the complete package.

Learn how to log onto Junosphere and load the topology file, then watch the book's videos, click through the topology animations, and then simply copy and paste from the PDF book's prompts to configure the Junosphere virtual machines online. You learn by doing, not reading.

Junosphere® provides a cost-effective and flexible environment where you can create and run networks in the cloud. These networks can be used for the same exercises you perform today in your physical lab and more, including network design, modeling, troubleshooting, testing, and training.

Virtual Day One - Learn by Doing!

- Build a BGP Multicast VPN using Junosphere
- Experience the signaling of Source-Specific Multicast (SSM) with BGP
- Watch Provider Tunnel Auto-Discovery
- Play with dynamic RSVP-TE Point-to-Multipoint MPLS Label Switched Paths
- Walk through the differences between Inclusive and Selective tunnels
- And much more in about 4.5 hours.

Published by Juniper Networks Books: http://www.juniper.net/books
Author and Video Editor: Antonio Sanchez-Monge
Video Narration: Dave Dugal
Technical Reviewers: Jeffrey (Zhaohui) Zhang, Aditya Mahale
Editor in Chief: Patrick Ames
Copyeditor and Proofer: Nancy Koerbel
J-Net Community Manager: Julie Wider

ISBN: 978-1-936779-61-1 (print)
Printed in the USA by Vervante Corporation, www.vervante.com

ISBN: 978-1-936779-62-8 (eBook)

Version History: v1 March 2013
2 3 4 5 6 7 8 9 10 #7100167-en

Acknowledgements

The author would like to thank: Patrick Ames, my faithful and enthusiastic companion in this adventure. Patrick's footprint is simply everywhere in *vDay One*: picking just two of the many highlights, his is the whole book design, as well as the idea of including links to videos. Dave Dugal, who happens to be one of my technical heroes, for his impressive voice and narration skills. Pilar Somohano, for her endless positive energy that inspired the idea of combining Junosphere with Day One. Oscar Pache for introducing me to Junosphere. Pablo Mosteiro and Anton Bernal for their unconditional support. Aleksey Mints, for his big help in the integration of the content. David Noguer, for his insight and energy in bringing this program to the market. Zoe Sands and Julie Wider, for their involvement in the marketing process. Jeffrey (Zhaohui) Zhang and Aditya Mahale for sharing their MVPN skills. Nikolay Manolov and Gonzalo Gomez for the route-prefix and bgp targets tips, respectively. Sergio Danelli, for the suggestion to use Cloud-shark. Domiciano Alonso, for sharing his technical insight during many years.

A big thanks to all the beta testers who provided feedback, helping to improve this content: from AT&T, Richard Duwi-quet; from Bell Canada, Eric Letourneau; from BT, Manuelo Orani, Matthias Blum, Brett Meyer, Andrew Riddell, Peter Pronk; from NYSE, Jacek Piterow, Lee Griggs, Derek Jobe; from Telefonica, Domiciano Alonso, Javier Santayana; from ONO, Ana Garcia Gonzalez, Jose Sanchez Chacon; from NSN, Carlos Saez, Daniel Fernandez, and Jose Manuel Perea; from Juniper Networks, Steven Wong, Josef Buchsteiner, Dan Rautio, Aditya Mahale, Vijay Kestur, Poorna Pushkala B, Sergio Danelli, Gonzalo Gomez, Nikolay Manolov, Diogo Montagner, Radu-Decebal Pavaloiu, Cantemir Olaru, Nicolas Marcoux, Andre Riscalla, Yuri Gordienko.

Welcome to *vDay One*

This *vDay One* book provides a virtual hands-on workshop with the following components:

- Videos: Each chapter contains a link to a YouTube video explaining the methodology or the relevant concepts in detail.
- A Real Scenario with Junos Routers: The topology used in this workshop is ready for you to start and it is in the Public Library of Junosphere.
- Packet Captures: All the captures highlighted in the videos, are available in Cloudshark to view and download.
- This Book: In order to keep you focused on the practical tasks, this book simply contains a step-by-step lab procedure, together with the pictures and the links to videos describing each scenario.

This *vDay One* book covers the fundamentals of BGP Multicast VPN. Since it's not possible to cover all the different variants of this technology in an introductory workshop, one transport flavor (RSVP-TE Point-to-Multipoint LSPs, with default template) and one service model (single-homed Source-Specific C-Multicast) are highlighted. The other flavors and more advanced configurations will be covered in future vDay One books.

Pre-requisites

The 4h30m of net time needed to go through the material on Junosphere is just an estimate. It is suggested that you book more time though to take breaks: you may be curious enough to check other commands, or you may need to spend additional time if you are new to Junosphere. The current reservation model in Junosphere works in a per-day basis, so it's flexible in that sense.

The pre-requisites for this virtual workshop are:

- A valid Junosphere account (http://www.junosphere.net). To order Junosphere with a special discount, go to https://learningportal.juniper.net/juniper/user_activity_info.aspx?id=5898 and enter promo code jun3928.
- You will need previous hands-on practice with the Junos OS CLI, and some knowledge about MPLS VPN and IP Multicast. Please visit the *Day One* books site (http://www.juniper.net/dayone) for introductory texts on Junos and the CLI. Other vDay One titles will be posted here, too.

Junosphere uses Network Connect to join the VM topology. Please note the following requirements:

- You need to have administration rights on your computer to install the Network Connect software.
- It is not possible to run two simultaneous instances of Network Connect, so if you are already have a Network Connect instance running for a corporate VPN, you will need to stop that first.
- Network Connect works best without web proxies, and it works fine with static proxy configuration as well. However, it doesn't work if the browser is configured with a PAC (Proxy Auto-Configuration) file.

TIP If you are using Windows, Google Chrome and Internet Explorer (in that order) are the preferred browsers. If you are using Mac OSX, go for Mozilla Firefox.

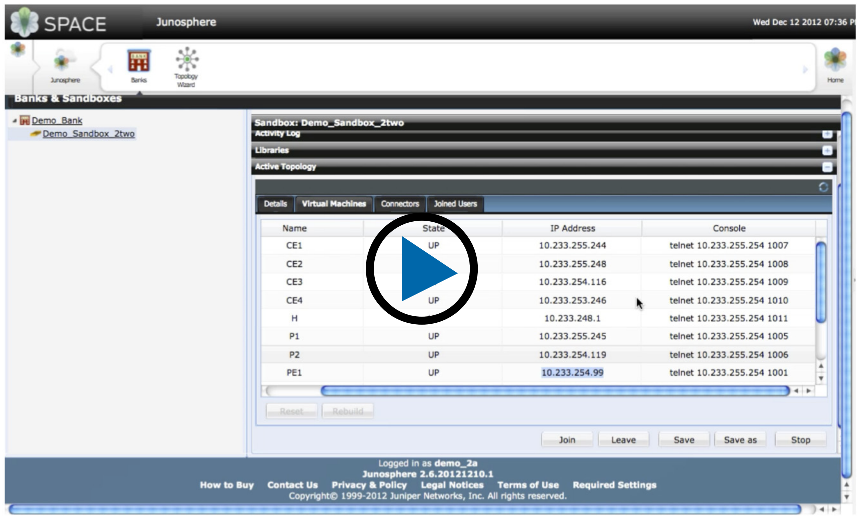
TIP If you'll be cutting and pasting config files directly from this PDF into the terminal, tests have shown using Acrobat Reader works better versus other apps with PDF capabilities – these other apps can run lines of code together.

1. Loading the Baseline Scenario

Follow the instructions in Video 1 to start the topology for *vDay One: Introduction to BGP MVPN*. You can find it in the Public Library called *Day One books* in Junosphere.

Video 1 also shows you how to download a file called *vDay_One_-_Introduction_to_BGP_MVPN.zip*, that you can examine if you are curious and want to understand some of the magic behind Junosphere. This zip file contains:

- A file entitled *topology.vmm*, which specifies the way the routers - or Virtual Machines - are interconnected.
- The initial Junos OS configuration of all the routers.



Name	State	IP Address	Console
CE1	UP	10.233.255.244	telnet 10.233.255.254 1007
CE2	[illegible]	10.233.255.248	telnet 10.233.255.254 1008
CE3	[illegible]	10.233.254.116	telnet 10.233.255.254 1009
CE4	UP	10.233.253.246	telnet 10.233.255.254 1010
H	[illegible]	10.233.248.1	telnet 10.233.255.254 1011
P1	UP	10.233.255.245	telnet 10.233.255.254 1005
P2	UP	10.233.254.119	telnet 10.233.255.254 1006
PE1	UP	10.233.254.99	telnet 10.233.255.254 1001

Logged in as **demo_2a**
Junosphere 2.6.20121210.1
How to Buy **Contact Us** **Privacy & Policy** **Legal Notices** **Terms of Use** **Required Settings**
Copyright© 1999-2012 Juniper Networks, Inc. All rights reserved.

Video 1 VM Topology Used in This Book

Start the topology using the instructions in Video 1, and verify that the *topology.vmm* file corresponds to Figure 1.

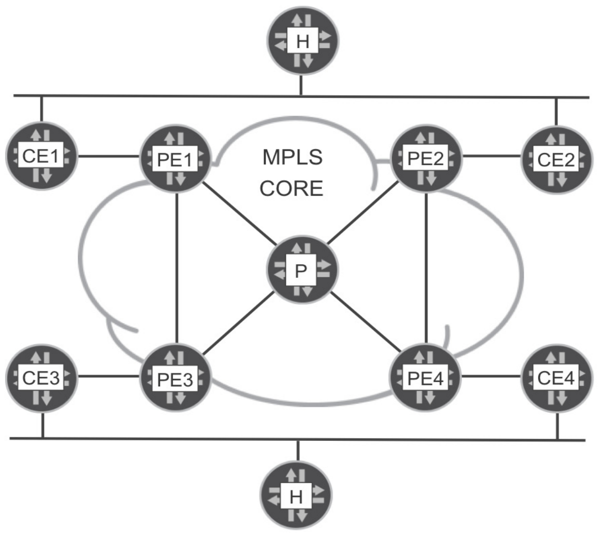


Figure 1 Physical Topology Used in This Book

2. BGP/MPLS VPN Refresher

In this section, you are going to follow 10.10.10/24, the unicast route to the C-Multicast source. This route is advertised throughout the network via BGP.

Open ten terminals, with one session each:

1. Session 1 connected to CE1
2. Session 2 connected to CE2
3. Session 3 connected to CE3
4. Session 4 connected to CE4
5. Session 5 connected to PE1
6. Session 6 connected to PE2
7. Session 7 connected to PE3
8. Session 8 connected to PE4
9. Session 9 connected to P
10. Session 10 connected to H

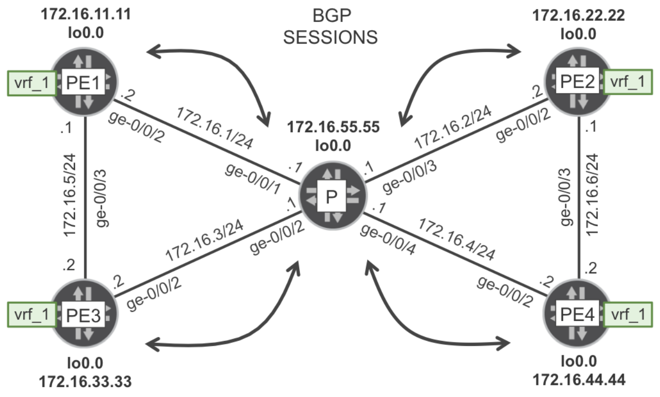


Figure 2 Logical Topology of the MPLS Core

Once the ten terminals are open, run the following on *all the devices*:

```
set cli screen-length 0
set cli screen-width 200
```

TIP You can copy the commands from this PDF and paste them directly into the terminals. With Adobe Acrobat Reader, several commands can be copied and pasted successfully in one go. Other PDF readers may merge several commands in one single line. Also, Adobe Acrobat Reader is proven to work fine with the video links in this book.

Then, verify that the MPLS core is healthy. At *all the PEs*, execute the following commands:

```
show isis adjacency
```

You'll see the two L2 IS-IS adjacencies are in the Up state.

Now on *all the PEs*:

```
show ldp neighbor
show ldp session
```

And there are two LDP neighbors and two open LDP sessions. Now *on any PE* you choose, have a look at the IS-IS database:

```
show isis database
```

You'll see Level 2 LSPs corresponding to P and to all the remote PEs. Now check what IS-IS routes have been installed in the master routing table `on` *any PE*:

```
show route protocol isis
```

And you can see all the remote core links and P/PE lo0.0 addresses. And finally *on any PE*:

```
show ldp database
```

Here is where you can find label bindings for all the lo0.0 addresses.

It's time to watch Video 2, and see the mechanics of unicast route signaling with BGP. To view, and optionally download, the packet captures referenced in Video 2, click on these links that will take you to cloudshark.org:

Capture 1
Capture 2
Capture 3

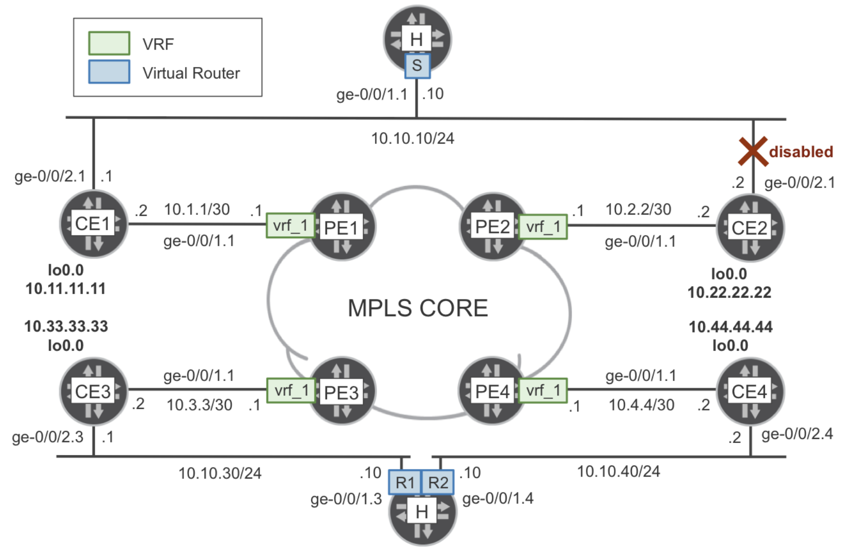
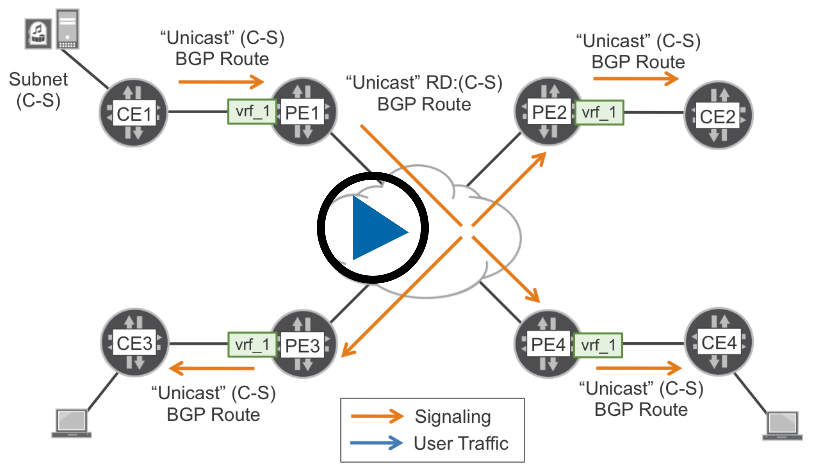


Figure 3 Logical Topology of the Access



Subnet (C-S)
"Unicast" (C-S) BGP Route
"Unicast" RD:(C-S) BGP Route
"Unicast" (C-S) BGP Route
CE1
vrf_1
PE1
PE2
vrf_1
CE2
CE3
vrf_1
PE3
PE4
vrf_1
CE4
"Unicast" (C-S) BGP Route
"Unicast" (C-S) BGP Route
Signaling
User Traffic

Video 2 Mechanics of Unicast route Signaling with BGP

TIP Want to take your own captures? Execute: `monitor traffic interface <interface> size 2000 no-resolve detail` . And you can use matching patterns: for example, BGP is selected with the `matching “port 179”` expression. Finally, if you want to save the capture to a file for offline viewing, you can (carefully) use the hidden and unsupported `write-file <file>` option.

Okay, now execute the following commands on *all the PEs:*

```
show bgp summary
```

Each PE learns a total of five BGP access routes. PE1 learns two routes from CE1, and three from the remote PEs, via the Route Reflector. PE2, PE3, and PE4 learn one route from the neighbor CE, and four from the remote PEs, via the RR. In this test topology, the RR function is performed by the P-router itself. This is not a common best practice because the RR should not be in the main forwarding path, but it’s acceptable for the purposes of this workshop.

Now check the BGP routes imported into the VRF *on all PEs*:

```
show route table vrf-1.inet.0 protocol bgp
```

You should see C-S/24 and the lo0.0 addresses of CE1,CE2,CE3, and CE4. Now, check what routes have been learned from CE1. Execute *at PE1*:

```
show route receive-protocol bgp 10.1.1.2 table vrf-1.inet.0
```

There are two routes: C-S/24 and the CE1 lo0.0. Now try *at PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.inet.0
```

And the routes learned from CE1 are re-advertised via IBGP to the Route Reflector. Execute *at PE3*:

```
show route receive-protocol bgp 172.16.55.55 table vrf-1.inet.0
```

The C-S/24 and the remote CE lo0.0 routes are received and installed. Now, while you are still *at PE3*:

```
show route advertising-protocol bgp 10.3.3.2 table vrf-1.inet.0
```

The routes learned from the Route Reflector are re-advertised to CE3. Now you can check the VRF policies *at any PE*:

```
show configuration routing-instances vrf-1
show configuration policy-options policy-statement vrf-1-in
show configuration policy-options policy-statement vrf-1-out
show configuration policy-options policy-statement ebgp-1-out
show configuration policy-options community unicast-comm-1
```

The `vrf-1-in` and `vrf-1-out` policies are dealing with the import and export of `inet-vpn unicast` (AFI=1, SAFI=128) routes from/to the Route Reflector. On the other hand, the `ebgp-1-out` policy controls the advertisement of the unicast routes to the neighboring CE.

Let’s follow the route again, all the way from CE1 to CE3, but this time looking at the extended communities. Let’s start *at PE1*:

```
show route receive-protocol bgp 10.1.1.2 table vrf-1.inet.0 extensive
```

There are no communities yet, but they are added in the next step. Still *at PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.inet.0 extensive
```

Now the route target community is added by policy `vrf-1-out`. Now, *move to PE3*:

```
show route receive-protocol bgp 172.16.55.55 extensive 10.10.10.0/24
```

The route is first imported in `bgp.l3vpn.0` (`inet-vpn` format), then in `vrf-1.inet.0` (`inet` format). Run *at PE3*:

```
show route advertising-protocol bgp 10.3.3.2 extensive 10.10.10.0/24
```

The prefix is advertised to CE3 with the route target community. Let's ping the future C-Multicast source *from CE3*:

```
ping count 1 10.10.10.10 source 10.33.33.33
```

It should be successful. Otherwise it's unexpected, and you would need to troubleshoot it further. Let's traceroute to follow the path to the C-Multicast Source from the loopback interface *at CE3*:

```
traceroute 10.10.10.10 source 10.33.33.33
```

The path goes through PE3 and PE1, bypassing P due to the IS-IS default metrics. Now let's follow the path to the other CEs *from CE3*:

```
traceroute 10.11.11.11 source 10.33.33.33
```

The path goes through PE3 and PE1, bypassing P due to the IS-IS default metrics. Again, *from CE3*:

```
traceroute 10.22.22.22 source 10.33.33.33
```

That path goes through PE3, P, and PE2. Execute *at CE3*:

```
traceroute 10.44.44.44 source 10.33.33.33
```

And the path goes through PE3, P, and PE4.

There is no point in sending the route target communities to the CEs. Configure the following *at all the PEs*:

```
configure
edit policy-options policy-statement ebgp-1-out term bgp-unicast
set then community delete unicast-comm-1
top
show | compare
commit and-quit
```

Check that the communities are correctly stripped *from PE3*:

```
show route advertising-protocol bgp 10.3.3.2 extensive 10.10.10.0/24
```

You'll see that the prefix is advertised to CE3 without the route target community.

TIP You can use the hidden (and, hence, unsupported) `show bgp targets` command during this and the next sections, for further insight on the role of route targets in the distribution of BGP routes.

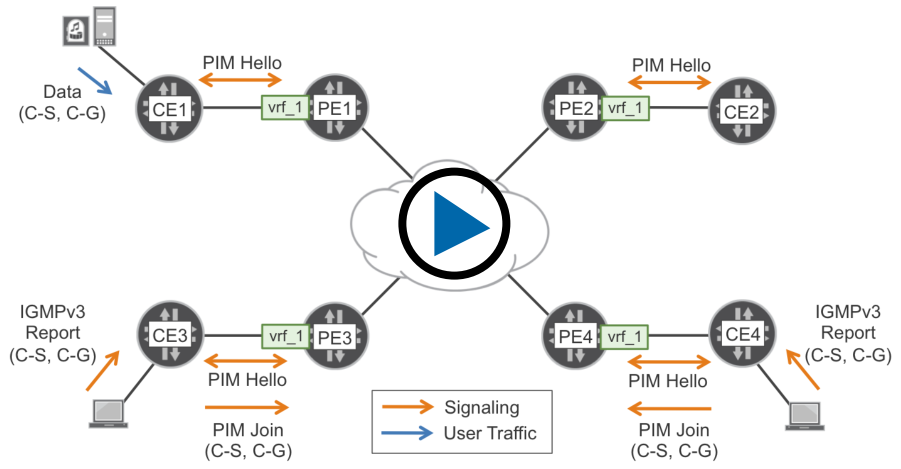
3. Source-Specific C-Multicast (SSM) at the Access

Watch Video 3 to refresh yourself on the basics of IPv4 Source-Specific Multicast (SSM).

To view and optionally download Packet Captures 4 and 5 referenced in Video 3, click on these links:

Capture 4

Capture 5



Video 3 **The Basics of IPv4 Source-Specific Multicast**

Start a C-Multicast flow *from H* and let it run (don't stop the command):

```
ping 239.1.1.1 ttl 10 interface ge-0/0/1.1 bypass-routing interval 0.1
```

IMPORTANT Make sure the ping command is running during the whole practice. If the session disconnects, start it again.

Execute *at CE1*:

```
show interfaces ge-0/0/2.1 statistics detail | match pps
```

If you see 0 pps, it's because PIM is not configured at the interface yet. Now configure PIM on the CE-PE link *at all the CEs*:

```
configure
set protocols pim interface ge-0/0/1.1 mode sparse
set protocols igmp interface ge-0/0/1.1 disable
show | compare
commit and-quit
```

IGMP is disabled here because you only needed it in the interfaces connected to C-Multicast receivers. It's okay if you don't disable it, too. In addition, configure PIM on the CE-to-S link *at CE1 and CE2 only*:

```
configure
set protocols pim interface ge-0/0/2.1 mode sparse
set protocols igmp interface ge-0/0/2.1 disable
show | compare
commit and-quit
```

And run this *on CE1*:

```
show interfaces ge-0/0/2.1 statistics detail | match pps
```

You'll see it's close to 10 pps, once PIM is configured at the interface.

NOTE This is also the reason why PIM is also enabled at R1 and R2 virtual routers within H. It has nothing to do with signaling, only with being able to count packets.

Configure PIM on the PE-CE interface *at all the PEs*:

```
configure
set routing-instances vrf-1 protocols pim interface ge-0/0/1.1 mode sparse
set protocols igmp interface ge-0/0/1.1 disable
show | compare
commit and-quit
```

PIM adjacencies should be up now. Execute *at all the PEs*:

```
show pim interface instance vrf-1
show pim neighbor instance vrf-1
```

LSI stands for Label Switched Interface, and is automatically created with the `vrf-table-label` configuration.

There should be one neighbor, the adjacent CE. Check the same at *all the CEs*:

```
show pim interface
show pim neighbor
```

There should be one neighbor, the adjacent PE. Now execute *at CE1*:

```
show multicast route extensive
```

Statistics should be close to 10pps, with the forwarding state as Pruned. For the moment, there are no active receivers. Check that on *CE3 and CE4*:

```
show pim join
```

It's empty because you didn't start the C-Multicast receivers yet. Let's now start them. Configure *two* subscriptions only *at CE3*:

```
configure
set protocols igmp interface ge-0/0/2.3 version 3
set protocols igmp interface ge-0/0/2.3 static group 239.1.1.1 source 10.10.10.10
set protocols igmp interface ge-0/0/2.3 static group 239.1.1.10 source 10.10.10.10
show | compare
commit and-quit
```

IGMP version 3 is used as it allows to define (S, G) subscriptions, as compared to IGMP version 2 that only supports (*, G). Now configure *one* subscription only *at CE4*:

```
configure
set protocols igmp interface ge-0/0/2.4 version 3
set protocols igmp interface ge-0/0/2.4 static group 239.1.1.1 source 10.10.10.10
show | compare
commit and-quit
```

Now let's check. Execute the following *at CE3 and CE4*:

```
show igmp group 239.1.1.1
```

And you'll see there are active subscriptions at both CE3 and CE4. Let's continue *at CE3 and CE4*:

```
show igmp group 239.1.1.10
```

There's an active subscription only at CE3. Now check PIM join state *at CE3*:

```
show pim join extensive
```

Look at the upstream and downstream interfaces: pseudo-GMP is related to IGMP subscriptions in an interface without PIM enabled. We will enable PIM here in a future vBook dealing about multicast redundancy. Let's check whether CE3 considers 10.10.10.10 as a reachable C-Multicast source, *at CE3*:

```
show pim source detail
```

Note that *active groups* do not mean there is actual multicast traffic. It is a signaling state. Check the multicast forwarding state, still *at CE3*:

```
show multicast route
```

It's empty. This is a cache normally filled with traffic, but there's no traffic yet. In order to spy on the PIM signaling, execute *at CE3*:

```
monitor traffic interface ge-0/0/1.1 matching pim detail
```

Let it run for a couple of minutes... and stop it with ctrl-C.

QUESTION 1 By default, how often are periodic PIM Hello and Join/Prune packets sent, respectively? The answer is at the end of the book.

After letting it run for a few minutes, run the following *on PE3 or PE4*:

```
show pim join instance vrf-1
show pim join instance vrf-1 extensive
```

On the upstream interface there's *no neighbor*. Let's see why. *On PE3 or PE4*:

```
show pim source instance vrf-1 detail
```

This is empty, because the C-Multicast source is not considered to be reachable yet. It is reachable from the unicast point of view, but the upstream router (PE1) is not a multicast neighbor. Again, *on PE3 or PE4*:

```
show multicast route instance vrf-1
```

Again, it's empty because there's no traffic yet in this side of the network. Finally, let's check the properties of the unicast route to the C-Multicast source *on PE3 or PE4*:

```
show route table vrf-1 10.10.10.10
show route table vrf-1 10.10.10.10 extensive | match comm
```

And only your custom route target shows.

4. Multicast VPN Site Auto-Discovery

In this section, you will see two major changes. First, the unicast routes will be modified in a crucial way, by adding two new extended communities. Second, the PEs will establish Multicast VPN neighbor relationships.

Refer to Table 1 during your remaining tasks.

Table 1 Multicast VPN (AFI=1, SAFI=5) BGP Route Types*

Route Type	Route Name	C-PIM equivalence	Functional Group
1	Intra-AS I-PMSI Auto-Discovery route	C-Hello	P-Route
2	Inter-AS I-PMSI Auto-Discovery route		
3	S-PMSI Auto-Discovery route		P-Route & C-Route
4	Leaf Auto-Discovery route		
5	Source Active Auto-Discovery route	(C-S, C-G) C-Register	C-Route
6	C-Multicast route - Shared Tree Join	(*, C-G) C-Join	
7	C-Multicast route - Source Tree Join	(C-S, C-G) C-Join	

* in this book only route types 1,3,4 and 7 are seen in action

To begin, let's enable the signaling of `inet-mvpn` routes. Configure *at P*:

```
configure
set protocols bgp group RR-CLIENTS family inet-mvpn signaling
show | compare
commit and-quit
```

And then *at all the PEs*:

```
configure
set protocols bgp group RR family inet-mvpn signaling
show | compare
commit and-quit
```

Execute *at PE1*:

```
show configuration protocols bgp
```

You'll see both address families `inet-vpn unicast` (AFI=1, SAFI=128) and `inet-mvpn` (AFI=1, SAFI=5). *Execute at PE1*:

```
show bgp summary
```

The new RIB bgp.mvpn.0, is still empty and still there's no MVPN route exchanged. Did the unicast route to the C-Multicast source change? *Execute at PE1*:

```
show route advertising-protocol bgp 172.16.55.55 extensive 10.10.10/24
```

Only your custom route target shows. The unicast (AFI=1, SAFI=128) route did not change. Let's check *on PE1* if there are MVPN adjacencies:

```
show mvpn neighbor instance-name vrf-1
```

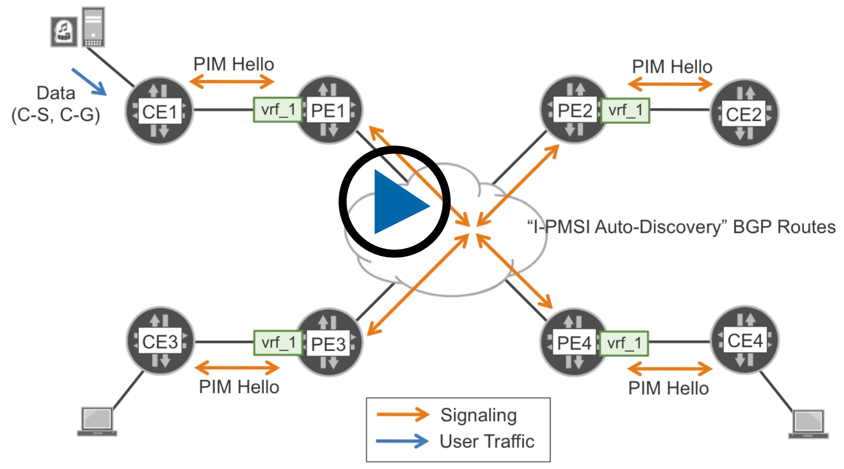
And MVPN is not running yet, so configure *on all the PEs*:

```
configure
set routing-instances vrf-1 protocols mvpn
show | compare
commit and-quit
```

Watch Video 4, to understand the implications of the configuration you just applied.

To view and optionally download Packet Captures 6, 7, and 8, referenced in Video 4, click on these links:

Capture 6
Capture 7
Capture 8



Video 4 Multicast VPN Site Auto-Discovery

TIP In the current version of Wireshark and in Cloudshark, some of the BGP attributes and RSVP objects described in this book may not be decoded yet by the current dissectors. Using Junos OS `tcpdump` at the routers may be an useful alternative. Just transfer them with binary FTP to one of the test routers (or VMs), and run:

```
start shell
% tcpdump -nvvr <capture_file>
```

Let's start by looking at the unicast (AFI=1, SAFI=128) routes. Execute *at PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.inet
```

You'll see the two usual unicast prefixes: C-S/24 and CE1 lo0.0 addresses. Now *at PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.inet.0 extensive
```

And here you'll see two new communities in addition to the route target: route import and source AS. Let's check the policies *at PE1*:

```
show policy
```

There are user-configured and also dynamic policies. *At PE1*:

```
show policy __vrf-mvpn-export-inet-vrf-1-internal__
```

This internal policy is responsible for adding these two communities. Now, let's see if these communities are received *at PE3*:

```
show route receive-protocol bgp 172.16.55.55 extensive 10.10.10.0/24
```

The C-S/24 route is received from the route reflector with the two new communities set at PE1. Are they sent to CE3? Execute *at PE3*:

```
show route advertising-protocol bgp 10.3.3.2 extensive 10.10.10.0/24
```

The src-as and rt-import communities are transitive, and propagated to the CEs.

QUESTION 2 How can you make sure that rt-import and src-as communities are not advertised to CEs? The answer is at the end of the book.

NOTE Look very carefully at Video 4 and at Capture 7. Do you notice a difference? Capture 7 has route import with community type 0x010a, while Video 4 shows 0x010b. By default, Junos uses 0x010a for backwards compatibility, but IANA later assigned 0x010b. In order to adapt to current IANA recommendations, you can configure: set protocols bgp group RR mvpn-iana-rt-import , then execute clear bgp neighbor 172.16.55.55. Even if you do it in one PE only, everything works fine since the implementation is flexible in route reception, allowing both 0x010a and 0x010b as valid community types.

You just saw how the unicast routes have been modified after enabling MVPN at the VRF. Let's now focus on the MVPN routes (AFI=1, SAFI=5). First, run the following *on PE1*:

```
show mvpn neighbor inet instance-name vrf-1
```

There are no neighbors. *On PE1*:

```
show route table vrf-1.mvpn.0
```

You'll only see the local I-PMSI Auto-Discovery route, but no remote routes, *on PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
```

This is empty. Okay, let's fix it by configuring the following *on all the PEs*:

```
configure
edit policy-options policy-statement vrf-1-out term multicast
set from family inet-mvpn
set then community add unicast-comm-1
set then accept
top
edit policy-options policy-statement vrf-1-in term multicast
set from family inet-mvpn
set from community unicast-comm-1
set then accept
top
show | compare
commit and-quit
```

And now check that the MVPN routes are now successfully exchanged, *at any PE*:

```
show bgp summary
```

There are three MVPN (AFI=1, SAFI=5) learned routes per VRF. One route per remote PE. They are installed first in `bgp.mvpn.0`, then in `vrf-1.mvpn.0`. Let's see what these routes accomplish, *at any PE*:

```
show mvpn neighbor inet instance-name vrf-1
```

You can see that all PEs are neighbors to each other, *at any PE*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
```

And the local site is announced. Now *at any PE*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn.0 extensive
```

The next hop is *irrelevant*. The only community included in this route is your custom route target for vrf-1. *At any PE*:

```
show route receive-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
```

The remote sites are discovered, *at any PE*:

```
show route receive-protocol bgp 172.16.55.55 table vrf-1.mvpn.0 extensive
```

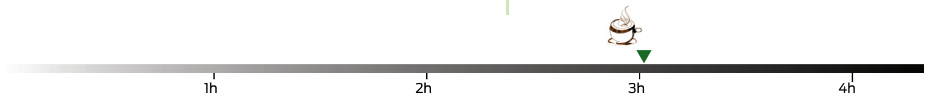
And again, you'll only see your custom route target for vrf-1. Finally, execute *at PE3*:

```
show route receive-protocol bgp 172.16.55.55 match-prefix 1:65000:1:172.16.11.11
```

You can use the `match-prefix` option throughout the book.

Coffee Break

Time for a coffee or some fresh water. Stand up and stretch. You've been multicasting for hours now.



5. C-Multicast SSM Signaling via BGP

The PEs are now completely empowered to signal C-Multicast state. However, PE3 and PE4 have not sent any Source Tree Join BGP route yet. Let's see why.

First, let's execute the following *at PE3*:

```
show pim join instance vrf-1
```

You'll see the upstream interface *Through BGP*. So far so good but let's see *at PE3* if the C-Multicast VPN state has been created.

```
show mvpn c-multicast inet instance-name vrf-1
```

For some reason, the join state was not propagated from PIM to MVPN. As a result, *at PE3*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn
```

Only the I-PMSI Auto-Discovery BGP route is being announced. In a pure SSM environment, the PIM (C-S, C-G) Joins will be converted to Source Tree Join BGP routes only if C-G is declared as a SSM group. And 239.1.1.1 is not included in the default SSM group range 232/8. So, let's configure the following *at all the PEs*:

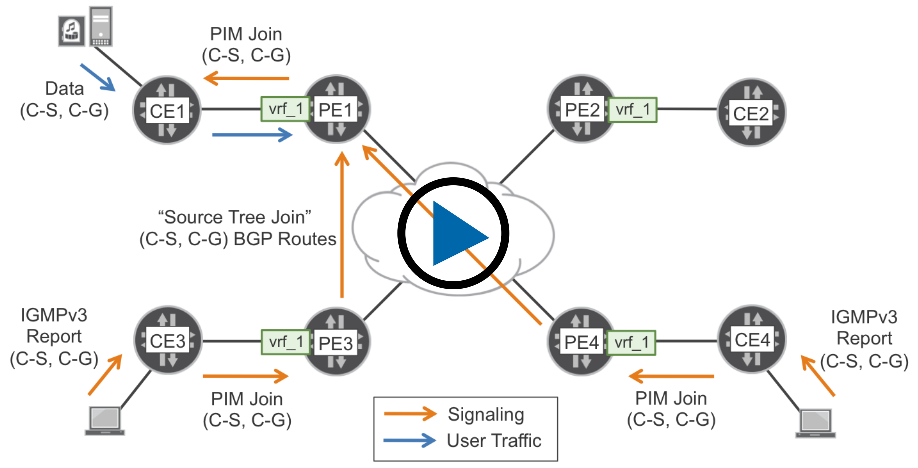
```
configure private
set routing-instances vrf-1 routing-options multicast ssm-groups 239.1/16
show | compare
commit and-quit
```

And in order to refresh the new SSM settings *on all the PEs*:

```
clear pim join instance vrf-1
```

Watch Video 5, to understand the new signaling triggered by the configuration you just applied.

To view and optionally download Packet Captures 9 and 10, referenced in Video 5, click on these links:
Capture 9
Capture 10



Video 5 Signaling Multicast Join State with BGP

Check the effect of the last configuration change *on PE3 and PE4*:

```
show mvpn c-multicast inet instance-name vrf-1
```

You can see that the SSM joins (two at PE3, one at PE4) are now part of MVPN state. Note that the Provider Tunnel column is empty. *On PE3 and PE4*:

```
show pim source instance vrf-1 resolve-mvpn-neighbor detail
```

And notice that the upstream neighbor is PE1. Still *at PE3 and PE4*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
```

The (C-S, C-G) BGP Source Tree Join routes (two from PE3, one from PE4) appeared. To check where these routes are targeted, let's execute *on all the PEs*:

```
show route receive-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
```

As you can see, the Source Tree Join routes are only imported in PE1. In order to see why, execute *at PE3 and PE4*:

```
show route table vrf-1.inet.0 10.10.10.10 extensive | match comm
```

Write down the *route import* community. You will need it later. *Now at PE3 and PE4*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn.0 extensive
```

And here you can see that the route *import* of the C-S unicast route is now used as the route *target* of the (C-S, C-G) Source Tree Join.

QUESTION 3 If PE1 has another VRF (vrf-2) with MVPN enabled, would the route import community of the routes in `vrf-2.inet.0` be the same as in `vrf-1.inet.0`? The answer is at the end of the book.

Let's see how the Source Tree Join BGP routes are propagated. Execute *at P*:

```
show route table bgp.mvpn
```

The route reflector receives Source Tree Join (10.10.10.10, 239.1.1.1) from both PE3 and PE4, and it selects one of these (identical) routes for reflection.

Now let's see how this route is imported *at PE1*:

```
show policy __vrf-mvpn-import-cmcast-vrf-1-internal__
show mvpn c-multicast inet instance-name vrf-1
```

Notice the RM flag. It means PE1 received a Source Tree Join from a remote PE. Now let's verify the conversion from MVPN to PIM join state *at PE1*:

```
show pim join instance vrf-1
show pim join instance vrf-1 extensive
```

Hmmmm, is there a downstream forwarding state? *At PE1*:

```
show multicast route instance vrf-1 extensive
```

Forwarding state? Non-zero pps for 239.1.1.1. Where is this traffic going? Let's look *on PE1*:

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

Next hop type of (C-S, C-G) route? Multicast discard. In order to complete the picture, execute *at CE1*:

```
show pim join extensive
show multicast route
show multicast route extensive | match pps
show route forwarding-table table default destination 239.1.1.1 extensive
```

And you'll see healthy multicast forwarding state. If you see the word "unicast" in this output, it's because of the internal terminology of next hop structures in Junos OS. Execute *at PE3 or PE4*:

```
show multicast route instance vrf-1
```

And it's empty, which is expected as PE1 is not injecting the traffic in the core yet.

6. Inclusive Provider Tunnels

You still need a mechanism to transport the C-Multicast traffic through the MPLS core. This is what you are about to enable.

To begin, let's execute *at PE1 only*:

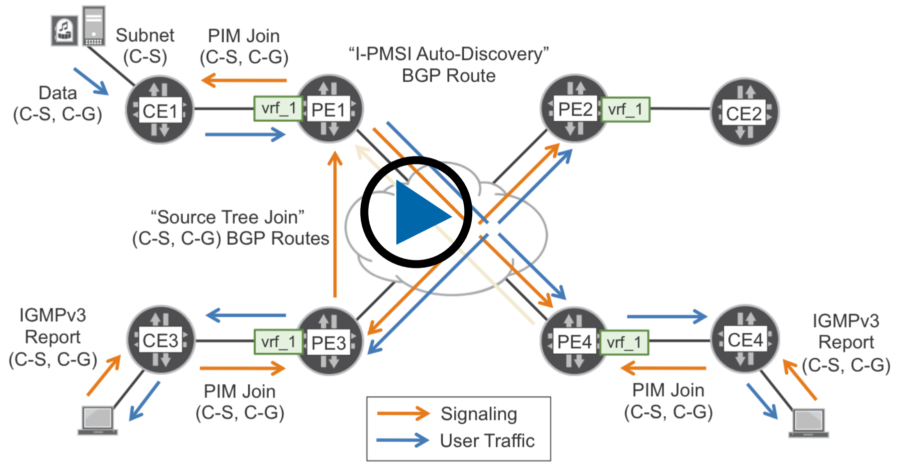
```
show rsvp session
```

It's empty, but not for a long time. The P-Tunnel flavor chosen in this task is RSVP-TE Point-to-Multipoint LSPs. So let's configure *at PE1*:

```
configure
edit routing-instances vrf-1
set provider-tunnel rsvp-te label-switched-path-template default-template
top
show | compare
commit and-quit
```

Watch Video 6, to understand the new signaling triggered by the configuration you just applied. To view and optionally download Packet Captures 11, 12, and 13, referenced in Video 6, click on these links:

Capture 11
Capture 12
Capture 13



Video 6 Inclusive Provider Tunnels

See how the previous configuration changed the MVPN route sent *by PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn extensive
```

There is a new attribute called PMSI. Write down the value of *alpha* in 172.16.11.11:0:*alpha*:172.16.11.11. This is the Tunnel ID that is also present in the RSVP session object, as you will see soon. *At PE1*:

```
show mvpn c-multicast inet instance-name vrf-1
```

Can you spot *alpha* here? *At PE1*:

```
show mvpn c-multicast inet instance-name vrf-1 display-tunnel-name
```

Note the provider tunnel name, and compare it with the next command output *at PE1*

```
show rsvp session p2mp
```

There are three sub-LSPs. Two of them (to PE2 and PE4) have the same label L0 (column Labelout), as expected. The remaining sub-LSP (to PE3) has label L3. Write down the numerical values of L0 and L3.

You already saw *alpha* in the BGP route. Let's now look for *alpha* in the RSVP context *at PE1*:

```
show rsvp session p2mp extensive | match port
```

Can you spot *alpha* here? Does it match the PMSI attribute of the I-PMSI Auto-Discovery BGP route? Good! That port number is included in the session RSVP object.

TIP In the sample packet captures 11, 12, and 13, the hexadecimal value of *alpha* was 0xd906. It is a dynamic value.

Execute *at PE1:*

```
show rsvp session p2mp statistics
```

Traffic should be increasing at around 10 packets per second (slightly less). *At PE1:*

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

Write down the values of L0 and L3, the labels to push towards P and PE3, respectively. Do they match the values found with `show rsvp session p2mp`? Now let's see how the C-Multicast traffic travels inside the P-Tunnel, *at PE1:*

```
show multicast route group 239.1.1.1 instance vrf-1 display-tunnel-name extensive
```

You can see the traffic is being forwarded down the Inclusive tunnel. Now let's focus on the interface statistics. *At PE1:*

```
show interfaces ge-0/0/2 statistics | match pps
show interfaces ge-0/0/3 statistics | match pps
```

Only one copy of each packet is sent out of each interface, as expected. Again *At PE1:*

```
show pfe statistics traffic | match pps
```

And there is more outgoing than incoming traffic, due to the replication! Let's move along the P-Tunnel. Execute *at router P*:

```
show route forwarding-table label <L0> extensive
```

Write down the values of L2 and L4, the labels to swap towards PE2 and PE4, respectively. Now check traffic replication:

```
show pfe statistics traffic | match pps
show interfaces ge-0/0/1 statistics | match pps
show interfaces ge-0/0/2 statistics | match pps
show interfaces ge-0/0/4 statistics | match pps
```

One copy of each packet is sent out of each output interface. P is replicating traffic.

NOTE If L2, L3, and L4 happen to be identical, it's due to the fact that each PE only has one VRF, and the label assigned to the first VRF is the same in all PEs. In production networks, the values will typically (but not necessarily) be different.

Let's look at the leaves of the P-Tunnel. Execute *at PE2*:

```
show route label <L2>
```

Note that table vrf-1 is shown as next hop. Let's examine the vrf-1 table *at PE2*:

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

The next hop is *discard*, as it has no downstream receivers. Still *at PE2*:

```
show multicast route group 239.1.1.1 instance vrf-1 extensive
```

And this shows that the traffic is arriving, but is locally discarded (Pruned). Execute *at PE3 and PE4*:

```
show rsvp session p2mp
```

Can you spot L3 and L4?

You already saw alpha in the BGP route as well as in the RSVP and MVPN contexts at PE1. Let's now look for alpha in the RSVP and MVPN contexts *at PE3 and PE4*:

```
show mvpn c-multicast inet instance-name vrf-1
show rsvp session p2mp extensive | match port
```

Now we focus on the forwarding plane, still *at PE3 and PE4*:

```
show route label <L3|L4>
```

Note that table vrf-1 is shown as next hop. Execute *at PE3 and PE4*:

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

The next hop is CE3|CE4, as expected. Still *at PE3 and PE4*:

```
show multicast route group 239.1.1.1 instance vrf-1 extensive
```

The traffic is being forwarded towards CE3 and CE4.Open a new terminal connected to H (don't stop the ping). Execute the following *at the host H*:

```
show interfaces ge-0/0/1.3 statistics detail | match pps
show interfaces ge-0/0/1.4 statistics detail | match pps
```

The traffic should arrive to both VLANs. Note that PIM is enabled in these interfaces with one single purpose: to be able to count the incoming multicast packets.

QUESTION 4 What would be the forwarding path for the flow (10.10.10.10, 239.1.1.*10*), as compared to the one you just saw for (10.10.10.10, 239.1.1.1) ? Give it a try!

Finally, in order to have a global view, let's execute the following *on all the PEs*:

```
show mvpn neighbor inet instance-name vrf-1
```

Only PE1 is a root of an Inclusive Provider Tunnel. This is fine as long as the C-Sources are behind PE1, which is the case here. Let's check the C-Multicast MVPN state *on all the PEs*:

```
show mvpn c-multicast inet instance-name vrf-1
```

It should match the current C-PIM join state of vrf-1 at each PE. Check at which PE the RM flag is set! And can you spot *alpha* here?

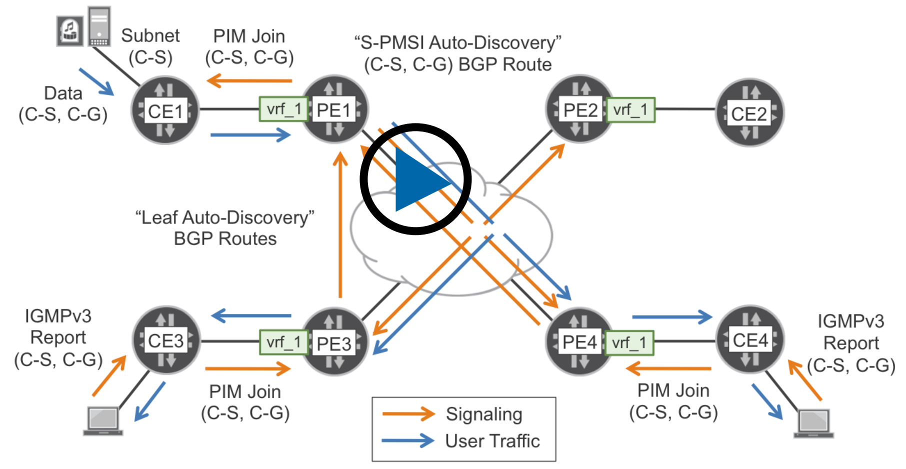
7. Selective Provider Tunnels

You already optimized traffic forwarding by using P2MP LSPs, however the C-Multicast traffic is reaching all the egress PEs, regardless of their downstream multicast state. Let's move one step forward in data plane optimization.

To begin, configure the following *at PE1*:

```
configure
edit routing-instances vrf-1 provider-tunnel selective group 239.1/16
set source 0/0 rsvp-te label-switched-path-template default-template
top
show | compare
commit and-quit
```

Watch Video 7, to understand the new signaling triggered by the configuration you just applied. Packet capture 14, referenced in Video 7 is available for download on cloudshark:
Capture 14



Video 7 Selective Provider Tunnels

Let's have a look at the new routes advertised *by PE1*:

```
show route advertising-protocol bgp 172.16.55.55 table vrf-1.mvpn extensive
```

You can see that the new Type 3 S-PMSI Auto-Discovery routes carry different PMSI values. For the route including (10.10.10.10, 239.1.1.1), write down *alpha1* as in 172.16.11.11:0:*alpha1*:172.16.11.11 . For the route including (10.10.10.10, 239.1.1.10), write down *alpha2* as in 172.16.11.11:0:*alpha2*:172.16.11.11.

Now let's see what routes have been received *by PE1*:

```
show route receive-protocol bgp 172.16.55.55 table vrf-1.mvpn.0
show route receive-protocol bgp 172.16.55.55 table vrf-1.mvpn.0 extensive | match comm
```

You should see two Leaf Auto-Discovery routes originated by PE3, and one Leaf A-D route originated by PE4. Now let's look at the MVPN C-Multicast state *at PE1*:

```
show mvpn c-multicast inet instance-name vrf-1
```

Can you spot *alpha1* and *alpha2* here? Let's move on. *At PE1*:

```
show mvpn c-multicast inet instance-name vrf-1 display-tunnel-name
```

Note the provider tunnel name, and compare it with the next command output *at PE1*:

```
show rsvp session p2mp
```

There are two new P2MP LSPs with 2 and 1 sub-LSPs, respectively, and the Labelout values towards P (L0) are different. Let's see the traffic statistics *at PE1*:

```
show rsvp session p2mp statistics
```

The P2MP LSP with 2 leaves shows traffic increasing at around 10 packets per second. Let's look for *alpha1* and *alpha2* in the RSVP context *at PE1*:

```
show rsvp session p2mp extensive | match port
```

Can you spot *alpha1* and *alpha2* here? Do they match the PMSI attribute of the BGP route above? Good!

TIP In the sample packet capture 14, the hexadecimal values of *alpha1* and *alpha2* are 0xd908 and 0xd909, respectively. These are dynamic values.

Now, let's follow the P-Tunnel. We start *at PE1*:

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

Write down the values of L0 and L3, the labels to push towards P and PE3, respectively. Let's check the interface statistics *at PE1*:

```
show interfaces ge-0/0/2 statistics | match pps
show interfaces ge-0/0/3 statistics | match pps
```

Only one copy of each packet is sent out of each interface. Let's move on *to P*:

```
show route forwarding-table label <L0> extensive
```

Write down the value of L4, the label to swap towards PE4. Still *at P*:

```
show interfaces ge-0/0/1 statistics | match pps
show interfaces ge-0/0/2 statistics | match pps
show interfaces ge-0/0/4 statistics | match pps
```

Here, no traffic is being sent to PE2, as expected, due to the selective nature of the tunnel. Execute the following *on PE3 and PE4*:

```
show route label <L3|L4>
```

Note that table vrf-1 is shown as next hop. Now *on PE3 and PE4*:

```
show route forwarding-table table vrf-1 destination 239.1.1.1 extensive
```

And here, the next hop is CE3|CE4, as expected. Still at *PE3 and PE4*:

```
show multicast route group 239.1.1.1 instance vrf-1 extensive
```

The traffic is being forwarded towards CE3 and CE4. Execute the following *at the host H*:

```
show interfaces ge-0/0/1.3 statistics detail | match pps
show interfaces ge-0/0/1.4 statistics detail | match pps
```

The traffic should arrive to both VLANs.

QUESTION 5 What would be the forwarding path for the flow (10.10.10.10, 239.1.1.*10*), as compared to the one you just saw for (10.10.10.10, 239.1.1.1) ? Give it a try!

Now, to have a global view, execute *on all the PEs*:

```
show mvpn c-multicast inet instance-name vrf-1
```

It should match the current C-PIM join state of vrf-1 at each PE. Can you spot *alpha1* and *alpha2* here?

8. Answers to the Questions in this Book

ANSWER 1 By default, PIM Hellos are sent every 30 seconds, and Join/Prune packets every 60 seconds.

ANSWER 2 Execute the following at *all the PEs*:

```
configure
edit policy-options community RT-IMPORT
set members rt-import:172.16.11.11:*
set members rt-import:172.16.22.22:*
set members rt-import:172.16.33.33:*
set members rt-import:172.16.44.44:*
up

set community SRC-AS members src-as:65000:0
set policy-statement ebgp-1-out term bgp-unicast then community delete RT-IMPORT
set policy-statement ebgp-1-out term bgp-unicast then community delete SRC-AS
show | compare
commit and-quit
```

ANSWER 3 No. The Route Import would be rt-import:172.16.11.11:N. The number *N* must differ between vrf-1 and vrf-2.

ANSWER 4 The new flow would use the same Inclusive Provider Tunnel, with exactly the same MPLS labels. It would reach PE2, PE3 and PE4. PE2 and PE4 would discard it, as they have no downstream receivers. Only PE3 would forward it down to CE3, and from CE3 it would reach R1 at H.

ANSWER 5 The new flow would use a different Selective Provider Tunnel, with different MPLS labels. This P2MP LSP only has one sub-LSP, whose destination is PE3. Only PE3 would receive and forward the traffic down to CE3, and from CE3 it would reach R1 at H.

MORE? Check out *This Week: Deploying BGP Multicast VPNs, 2nd Edition*, on the Day One web site at http://www.juniper.net/dayone.

Juniper Networks Junosphere cloud-based services allow networking professionals to perform network testing, design, and training exercises in a risk-free virtual environment that uses real network operating systems. Junosphere allows you to closely replicate physical networks consisting of Junos OS-based devices and ecosystem tools without the cost, complexity, or limitations of a physical lab.

To ensure you have the best possible experience with Junosphere, check that you have required settings. Consider these recommendations for optional freeware programs to facilitate Junosphere usage.

Required Settings	■ Only Mozilla 3.0-5.0 and Internet Explorer 7.0 -8.0 is supported ■ Enable pop-ups for junosphere.net ■ Allow downloads from junosphere.net ■ Install Java plug-in
Recommended Downloads	■ *RealVNC* - Remote access to the CentOS server ■ *PuTTY* - SSH/telnet client to access device consoles ■ *Notepad++* - Reader of configuration files ■ *FileZilla* - FTP client to access device consoles ■ *7zip* - Creates compressed topology filesets ■ *VMWare player* - To run the connector

Client Hardware Recommendations

CPU: 1 GHz or higher is recommended for Windows; for Mac, 1 GHz G4 or Intel processor is recommended.

Memory: Minimum of 256 MB of available RAM is recommended.

Color quality: For best results, use 16-bit (8-bit, 24-bit, and 32-bit are also supported).

Monitor resolutions: 1,024 x 768 pixels is recommended; up to 2,048 x 2,048 pixels is supported.

PDF Recommendations

Use Acrobat Reader to copy and paste this book's config files into the terminal for the best results.

Check for the most recent updates and specifications at www.juniper.net/junosphere.

junosphere

ISBN 978-1936779611
50900
9 781936 779611
07100167

JUNIPER NETWORKS