

DAY ONE: NORTHSTAR CONTROLLER UP AND RUNNING

Introduce yourself to the NorthStar Controller by focusing on the discovery and visualization of IP/MPLS networks – you'll be amazed by the centralized visibility into your entire network.

By Patricio Giecco

DAY ONE: NORTHSTAR CONTROLLER UP AND RUNNING

Day One: NorthStar Controller Up and Running is intended for all networking professionals working on WAN and LAN environments that make use of IP/MPLS services. It should also be of interest to high-level technical professionals looking to understand the fundamentals of Juniper's approach to Software Defined Networks (SDN) in MPLS transport networks.

The book introduces readers to the Juniper NorthStar Controller by focusing on the discovery and visualization of IP/MPLS networks including the ability to visualize the paths different LSPs take on the network, monitoring the status and utilization of the network in real-time, and, modeling the impact of network changes, among other use cases.

"Modern networks are increasingly in need of more sophisticated traffic-engineering services while at the same time the management of even the most basic traffic engineering requires simplification. The Juniper NorthStar Controller provides the necessary tools to deliver just that! With its programmable and open framework, it offers Service Providers, Content Providers, and Enterprises a simple, yet sophisticated toolset to take control of their network by optimizing their existing infrastructure while driving new services."

Colby Barth, Distinguished Engineer, Juniper Networks, Inc.

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Discover the network topology by using OSPF or IS-IS to peer with the network
- Use BGP-LS to extract the traffic engineering database from some nodes in an IP/MPLS network
- Configure PCEP to extract LSP information, obtain notifications and modify or provision LSPs
- Monitor the status of an MPLS network in real-time
- Visualize various aspects of the network such as its topology, utilization, and path placement
- Utilize the controller to create and modify LSPs
- Delegate control of existing LSPs to the NorthStar Controller

Juniper Networks Books are singularly focused on network productivity and efficiency. Peruse the complete library at www.juniper.net/books.

Published by Juniper Networks Books



JUNIPER
NETWORKS

Juniper Networking Technologies

Day One: NorthStar Controller Up and Running

By Patricio Giecco

<i>Chapter 1: Getting Started</i>	<i>7</i>
<i>Chapter 2: Using the NorthStar Controller</i>	<i>29</i>
<i>Chapter 3: MPLS LSP Management.....</i>	<i>47</i>
<i>Chapter 4: Troubleshooting</i>	<i>63</i>
<i>Appendix.....</i>	<i>71</i>

© 2015 by Juniper Networks, Inc. All rights reserved. Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, 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.

Published by Juniper Networks Books

Authors: Patricio Giecco

Technical Reviewers: Colby Barth, Naresh Kumar,

Rendo Wibawa, Tony Le

Editor in Chief: Patrick Ames

Copyeditor and Proofer: Nancy Koerbel

Illustrator: Karen Joice

J-Net Community Manager: Julie Wider

ISBN: 978-1-941441-31-2 (print)

Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-32-9 (ebook)

Version History: v1, December 2015

2 3 4 5 6 7 8 9 10

About the Author

Patricio Giecco is a Product Architect who has worked in a number of positions in the Networking industry, including Solutions Engineering, Technical Marketing, and Product Management. In his tenure at Juniper, Patricio has written many application notes, co-authored *Junos Security* (O'Reilly Media, 2010), and was a recipient of a Juniper Star award in 2011 and 2014.

Author's Acknowledgments

I would like to thank David Ranch and Anand Athreya, both of whom have shown incredible patience in having to deal with an outspoken Argentinean with an uncanny ability to always say the wrong things.

This book is available in a variety of formats at:

<http://www.juniper.net/dayone>.

Welcome to Day One

This book is part of a growing library of *Day One* books, produced and published by Juniper Networks Books.

Day One books were conceived to help you get just the information that you need on day one. The series covers Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

The *Day One* library also includes a slightly larger and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a weeklong seminar.

You can obtain either series, in multiple formats:

- Download a free PDF edition at <http://www.juniper.net/dayone>.
- Get the ebook edition for iPhones and iPads from the iTunes Store. Search for Juniper Networks Books.
- Get the ebook edition for any device that runs the Kindle app (Android, Kindle, iPad, PC, or Mac) by opening your device's Kindle app and going to the Kindle Store. Search for Juniper Networks Books.
- Purchase the paper edition at either Vervante Corporation (www.vervante.com) for between \$12-\$28, depending on page length.

Audience

This book is intended for all networking professionals working on WAN and LAN environments that make use of IP/MPLS services. You should be at least at an intermediate level in your knowledge of, and skills in, working with devices running the Junos OS.

In addition, the book may be of interest to high-level technical professionals looking to understand the fundamentals of Juniper's approach to Software Defined Networks (SDN) in MPLS transport networks

MORE?

It's highly recommended you read through the technical documentation and get a sense of the NorthStar Controller before you jump in. Try the *NorthStar Controller Getting Started Guide* at the Juniper TechLibrary: http://www.juniper.net/techpubs/en_US/northstar1.0.0/information-products/pathway-pages/getting-started.html.

What You Need to Know Before Reading This Book

Since we will be looking at IP/MPLS networks that make extensive use of traffic engineering, the reader is expected to be familiar with the protocols underpinning these networks. A basic understanding of the following protocols is assumed:

- OSPF and IS-IS and how these protocols propagate topology information
- RSVP-TE
- Basic understanding of CSPF and how MPLS networks compute paths
- Some familiarity with the contents and uses of the traffic engineering database (TED) is useful, but not required.

What You Will Learn by Reading This Book

- Discover the network topology by using OSPF or IS-IS to peer with the network.
- Use BGP-LS to extract the traffic engineering database from some nodes in an IP/MPLS network.
- Configure PCEP to extract LSP information, obtain notifications and modify or provision LSPs.
- Monitor the status of an MPLS network in real-time.
- Visualize various aspects of the network such as its topology, utilization, and path placement.
- Delegate control of existing LSPs to the NorthStar controller.
- Utilize the controller to create and modify LSPs.
- Optimize the paths the LSPs take in the network according to use-configurable parameters and constraints.
- Model the impact of network changes and how those changes would affect the placement of LSPs and the utilization of the various links.
- Provision new LSPs using a graphical topology in a simple manner. Provision a large number of LSPs following traditional full-mesh or hub-and-spoke topologies in a few clicks.

Chapter 1

Getting Started

Proper management of IP/MPLS networks requires the ability to constantly monitor the status of different aspects of the network. When dealing in particular with MPLS transport networks, it's necessary to pay special attention to not only the status of the different LSPs used to carry traffic, but also how the different protection mechanisms are configured, how much coverage those mechanisms bring to different failure scenarios, and, how optimal the routing of the LSPs happens to be. So this *Day One* book focuses on how to use NorthStar to monitor and visualize the status of an IP/MPLS network.

Many different mechanisms are already commonly used to obtain operational information from a network, including:

- Using SNMP to poll data from the different network devices.
- Receiving SNMP traps and other events (such as syslog messages) from the network devices.
- Using the device's out-of-band configuration management mechanisms, like NETCONF or the CLI, to obtain the configuration information of the various devices in the network.
- Using NETCONF and the CLI to obtain operational status of the network.

While extremely powerful, most of these mechanisms suffer from several shortcomings. For example, using the device's CLI to fetch configuration information from the different devices. While the CLI can provide a lot of information, it is a processing intensive operation that requires the ability to connect to the different devices, load and parse the configurations, and then build a network model from the information discovered. For medium-to-large networks, the whole process can take minutes, so it can't be done in real time (even when the mechanisms to synchronize configuration changes with the management system *do* exist).

On the other hand, SNMP traps and events can inform the management systems of changes in real-time, but complex event correlation rules are needed in order to find out which elements and services in the network are affected. In practice, not only does this require a number of different mechanisms to be used simultaneously but it can also add significant delays between the time the network changes and when the management system is actually updated of the changes.

Consider, for example, the monitoring of the status of some LSPs in the network. This requires knowledge of not only the network's topology, but also some visibility into the path taken by all the LSPs being monitored in the network. It's clear to most engineers that obtaining this information using traditional mechanisms is a time-consuming task. Furthermore, if sudden network changes modify the existing topology, the management station needs to correlate any failures with the affected LSPs and poll the network, again, to obtain new path information.

Better mechanisms are needed to obtain up-to-date information of the status of today's modern IP/MPLS networks. And this book will show you that better mechanism. In fact, you are going to build it.

Let's start with the test or lab network.

Building a Model Network

A basic network topology is used to construct this book's different use cases and examples. This topology is the result of some of the work done in the Juniper Networks Proof of Concept (POC) Lab in Sunnyvale, California, as well as with various demo systems build with the Juniper NorthStar Controller product team, to field-demonstrate some of the controller's features. A few changes were made to the topology in order to simplify the configuration for the sake of brevity and also so the book's tutorials don't have to spend time and space explaining configurations that aren't pertinent to the *Day One* task.

The topology contains a few PE routers where our LSPs initiate, and also where, presumably, upper layer services that make use of the LSPs are configured.

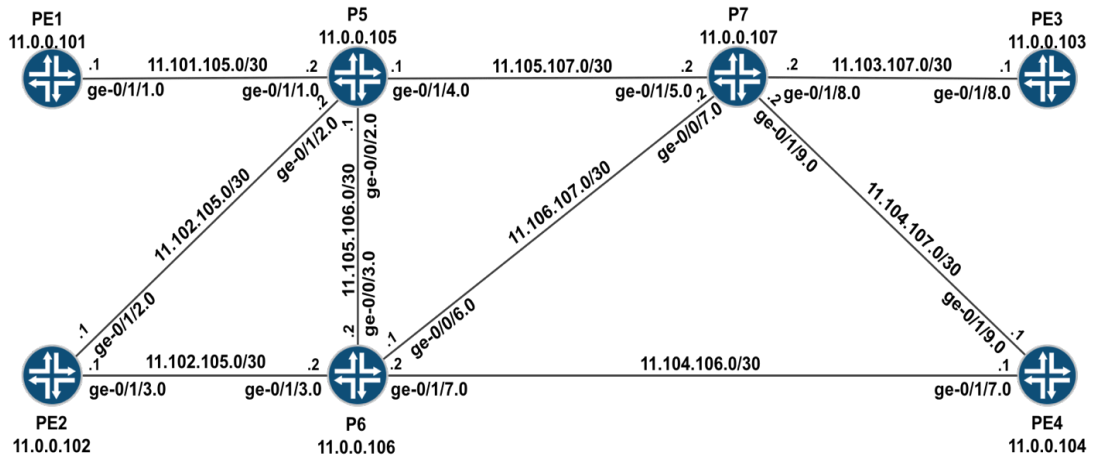


Figure 1.1 The Book's Model Network

In the book's lab, the full topology was built using virtual MX (vMX) devices because of its convenient way to test and build new topologies without having to do any cabling or racking. It has no impact on the configurations whatsoever. We recommend you try it in your own lab.

What will make the configurations slightly different for you is that the book's test bed used logical systems to simulate two of the P routers, which reduced the total number of vMXs required to build the topology. This book shows the configurations of the devices ignoring that logical systems were used, so they can be used without any modifications in a network with seven devices (physical or virtual). The full configuration of the vMX devices for the baseline network is listed in the Appendix, so you can use it to recreate the exact topology, if you so desire. Otherwise the relevant configuration snippets are included as the book goes through the use cases.

Okay, enough about the lab. To get started you need to configure the usual routing and signaling protocols required for RSVP-TE to work, namely OSPF with traffic engineering extensions, and RSVP must be enabled.

You can skim through the configuration to get all the details but these are the main characteristics of our model network:

- OSPF is enabled in all interfaces, including the loopback. All interfaces are part of area 0.

- Traffic engineering extensions have been enabled on OSPF, so the traffic engineering database used by RSVP can be built.
- Since all the devices are connected back-to-back using Ethernet interfaces, the interface types have been set to P2P.
- RSVP is enabled in all interfaces.
- MPLS is configured in all interfaces.
- Some LSPs are provisioned, which are used later on when the book gets to LSP visualization and path computation.
- Some SRLGs are also configured, which can be used when testing path computation calculations and LSP placement. (This step is completely optional.)

```

PE1
/* General settings */
set system host-name PE1

/* Interface configuration */
set interfaces ge-0/1/1 unit 0 family inet address 11.101.105.1/30
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.101/32 primary
set interfaces lo0 unit 0 family mpls

/* Enable RSVP */
set protocols rsvp interface all bandwidth 40g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/1.0 bandwidth 40g

/* Basic routing options, some of these settings are used in some devices in the BGP
configuration scenarios */
set routing-options router-id 11.0.0.101
set routing-options autonomous-system 100
/* We'll setup some SRLGs so when we play with PATH computations, we can use SRLGs to
influence the path calculation */
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50

/* MPLS is also enabled in all interfaces, except for the management interface */
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/1/1.0 srlg srlg-100
/* Add some LSPs to the network for later when visualizing LSPs and their paths */
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE1_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE1_PE2 bandwidth 500m
set protocols mpls label-switched-path LSP_PE1_PE2 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE2 adaptive
set protocols mpls label-switched-path LSP_PE1_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE1_PE3 bandwidth 500m
set protocols mpls label-switched-path LSP_PE1_PE3 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE3 adaptive
set protocols mpls label-switched-path LSP_PE1_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE1_PE4 bandwidth 500m

```

```
set protocols mpls label-switched-path LSP_PE1_PE4 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE4 adaptive
/* OSPF configuration */
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all interface-type p2p

PE2
set system host-name PE2
set interfaces ge-0/1/2 unit 0 family inet address 11.102.105.1/30
set interfaces ge-0/1/2 unit 0 family mpls
set interfaces ge-0/1/3 unit 0 family inet address 11.102.106.1/30
set interfaces ge-0/1/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.102/32
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options router-id 11.0.0.102
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface ge-0/1/3.0 bandwidth 40g
set protocols rsvp interface fxp0.0 disable
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE2_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE2_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE2_PE1 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE1 adaptive
set protocols mpls label-switched-path LSP_PE2_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE2_PE3 bandwidth 500m
set protocols mpls label-switched-path LSP_PE2_PE3 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE3 adaptive
set protocols mpls label-switched-path LSP_PE2_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE2_PE4 bandwidth 500m
set protocols mpls label-switched-path LSP_PE2_PE4 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE4 adaptive
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/1/2.0 srlg srlg-100
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable

PE3
set system host-name PE3
set interfaces ge-0/1/8 unit 0 family inet address 11.103.107.1/30
set interfaces ge-0/1/8 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.103/32
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options router-id 11.0.0.103
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/8.0 bandwidth 40g
```

```

set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE3_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE3_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE1 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE1 adaptive
set protocols mpls label-switched-path LSP_PE3_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE3_PE2 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE2 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE2 adaptive
set protocols mpls label-switched-path LSP_PE3_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE3_PE4 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE4 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE4 adaptive
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable

```

PE4

```

set system host-name PE4
set interfaces ge-0/1/7 unit 0 family inet address 11.104.106.1/30
set interfaces ge-0/1/9 unit 0 family inet address 11.104.107.1/30
set interfaces lo0 unit 0 family inet address 11.0.0.104/32 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options router-id 11.0.0.104
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/7.0 bandwidth 40g
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE4_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE4_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE1 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE1 adaptive
set protocols mpls label-switched-path LSP_PE4_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE4_PE2 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE2 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE2 adaptive
set protocols mpls label-switched-path LSP_PE4_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE4_PE3 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE3 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE3 adaptive
set protocols mpls interface all
set protocols mpls interface ge-0/1/9.0 srlg srlg-407
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable

```

P5

```

set system host-name P5
set interfaces ge-0/0/2 unit 0 family inet address 11.105.106.1/30
set interfaces ge-0/0/2 unit 0 family mpls

```

```
set interfaces ge-0/0/4 unit 0 family inet address 11.105.107.1/30
set interfaces ge-0/0/4 unit 0 family mpls
set interfaces ge-0/1/1 unit 0 family inet address 11.101.105.2/30
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces ge-0/1/2 unit 0 family inet address 11.102.105.2/30
set interfaces ge-0/1/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.105/32
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options router-id 11.0.0.105
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/1.0 bandwidth 40g
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

P6

```
set system host-name P6
set interfaces ge-0/0/3 unit 0 family inet address 11.105.106.2/30
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces ge-0/0/6 unit 0 family inet address 11.106.107.1/30
set interfaces ge-0/0/6 unit 0 family mpls
set interfaces ge-0/1/3 unit 0 family inet address 11.102.106.2/30
set interfaces ge-0/1/3 unit 0 family mpls
set interfaces ge-0/1/7 unit 0 family inet address 11.104.106.2/30
set interfaces ge-0/1/7 unit 0 family mpls
set interfaces lo0 unit 106 family inet address 11.0.0.106/32
set interfaces lo0 unit 106 family mpls
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/3.0 bandwidth 40g
set protocols rsvp interface ge-0/1/7.0 bandwidth 40g
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
```

P7

```
set interfaces ge-0/0/5 unit 0 family inet address 11.105.107.2/30
set interfaces ge-0/0/5 unit 0 family mpls
set interfaces ge-0/0/7 unit 0 family inet address 11.106.107.2/30
set interfaces ge-0/0/7 unit 0 family mpls
set interfaces ge-0/1/8 unit 0 family inet address 11.103.107.2/30
set interfaces ge-0/1/8 unit 0 family mpls
set interfaces ge-0/1/9 unit 0 family inet address 11.104.107.2/30
set interfaces ge-0/1/9 unit 0 family mpls
set interfaces lo0 unit 107 family inet address 11.0.0.107/32
set interfaces lo0 unit 107 family mpls
set protocols rsvp interface all bandwidth 10g
```

```

set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/8.0 bandwidth 40g
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/0/9.0 srlg srlg-407
set protocols mpls interface ge-0/1/9.0 srlg srlg-407
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50

```

After the network is set up, you should be able to see all devices and links in the traffic engineering database (TED). Let's check the contents of the TED, to make sure all nodes and links are present (the contents should be the same in all nodes):

```

root@P3# run show ted database
TED database: 0 ISIS nodes 7 INET nodes
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.101                       Rtr   660      1      1 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.101.105.1, Remote: 11.101.105.2
  Local interface index: 334, Remote interface index: 0
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.102                       Rtr   669      2      2 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.102.105.1, Remote: 11.102.105.2
  Local interface index: 334, Remote interface index: 0
  To: 11.0.0.106, Local: 11.102.106.1, Remote: 11.102.106.2
  Local interface index: 335, Remote interface index: 0
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.103                       Rtr    68      1      1 OSPF(0.0.0.0)
  To: 11.0.0.107, Local: 11.103.107.1, Remote: 11.103.107.2
  Local interface index: 332, Remote interface index: 0
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.104                       Rtr   101      2      2 OSPF(0.0.0.0)
  To: 11.0.0.107, Local: 11.104.107.1, Remote: 11.104.107.2
  Local interface index: 334, Remote interface index: 0
  To: 11.0.0.106, Local: 11.104.106.1, Remote: 11.104.106.2
  Local interface index: 333, Remote interface index: 0
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.105                       Rtr   666      4      4 OSPF(0.0.0.0)
  To: 11.0.0.101, Local: 11.101.105.2, Remote: 11.101.105.1
  Local interface index: 346, Remote interface index: 0
  To: 11.0.0.107, Local: 11.105.107.1, Remote: 11.105.107.2
  Local interface index: 340, Remote interface index: 0
  To: 11.0.0.106, Local: 11.105.106.1, Remote: 11.105.106.2
  Local interface index: 338, Remote interface index: 0
  To: 11.0.0.102, Local: 11.102.105.2, Remote: 11.102.105.1
  Local interface index: 347, Remote interface index: 0
ID                               Type Age(s) LnkIn LnkOut Protocol
11.0.0.106                       Rtr   224      4      4 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.105.106.2, Remote: 11.105.106.1
  Local interface index: 339, Remote interface index: 0
  To: 11.0.0.104, Local: 11.104.106.2, Remote: 11.104.106.1
  Local interface index: 349, Remote interface index: 0

```

```
To: 11.0.0.107, Local: 11.106.107.1, Remote: 11.106.107.2
  Local interface index: 342, Remote interface index: 0
To: 11.0.0.102, Local: 11.102.106.2, Remote: 11.102.106.1
  Local interface index: 348, Remote interface index: 0
ID                                     Type Age(s) LnkIn LnkOut Protocol
11.0.0.107                           Rtr   100    4    4 OSPF(0.0.0.0)
To: 11.0.0.105, Local: 11.105.107.2, Remote: 11.105.107.1
  Local interface index: 341, Remote interface index: 0
To: 11.0.0.104, Local: 11.104.107.2, Remote: 11.104.107.1
  Local interface index: 351, Remote interface index: 0
To: 11.0.0.106, Local: 11.106.107.2, Remote: 11.106.107.1
  Local interface index: 343, Remote interface index: 0
To: 11.0.0.103, Local: 11.103.107.2, Remote: 11.103.107.1
  Local interface index: 350, Remote interface index: 0
```

If some of the nodes or links are missing from your TED, you’ll need to start looking into the configurations of the individual protocols (OSPF, RSVP, and MPLS) and interfaces to make sure everything is in order.

NOTE In many ways it doesn’t matter what your lab network topology is, nor what the details regarding the metrics and bandwidths are. Most of the examples in this book can be transposed to any lab topology you currently have running and the reference network can be used to follow along more easily.

That’s the test network used for this book. Now it’s time for the star of the show, the NorthStar Controller.

Installing the NorthStar Controller

A fresh software install is a pretty straightforward process: just insert a CD or USB flash drive with the NorthStar installer and the installation process takes care of building the server, installing the OS, and all the other required packages.

The NorthStar software runs on 64-bit x86 processors and requires a minimum 16G of RAM to run. For a lab setup (unsupported), a small server will do, but in a production environment, the minimum hardware recommendation listed in Table 1.1 should be used.

Table 1.1 Minimum Northstar Hardware Recommendations

Deployment	CPU	RAM	Hard Drive	Network
Small (1-50 nodes)	64-bit Dual 1.8GHz Intel Xeon E5 equivalent	16 GB	250 GB	2x 1/10 GE
Medium (50-250 nodes)	64-bit Quad 2.26 GHz Intel Xeon E5220 equivalent	64 GB	500 GB	2x 1/10 GE
Large (more than 250 nodes)	64-bit Quad 2.93 GHz Intel Xeon X5570 equivalent	128 GB	1 TB	2x 1/10 GE

The install package includes all the services and database backend as well as the Junos virtual machine used for the topology acquisition.

1. Download the NorthStar ISO file from the support site. <http://www.juniper.net/support/downloads/?p=northstar>

2. Create a bootable USB drive or CD from the ISO file, and use your preferred utility to do this. For example, in machines running most Linux distributions you can use the `dd` tool to create a bootable USB drive.

MORE? You can find detailed instructions on how to build a bootable USB drive from an ISO file here: https://wiki.archlinux.org/index.php/USB_flash_installation_media.

3. Boot the server from the newly created CD or USB drive (you may have to change the BIOS settings in your machine so it will boot from the USB/CD drive). The boot loader will show you a few options, select the install from CDROM or USB. If you have a keyboard and monitor connected to the server, you can use the interactive setup option. Otherwise, simply choose the install using a serial port.

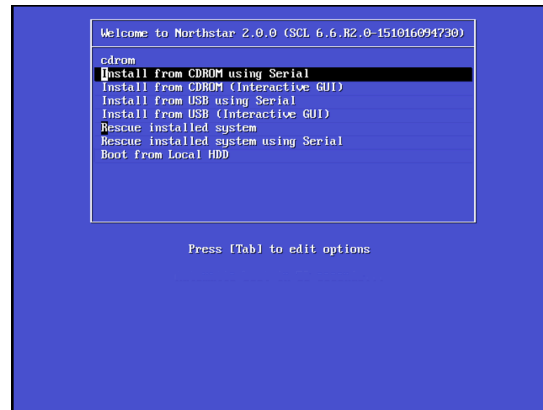


Figure 1.2 Boot Loader

4. The install process is largely automatic and at the end it will shut down the server. Make sure you remove the USB or CD drive *before* you power the server on again, otherwise the installer will be run again. Here, it's coming online:

```
Greetings.
anaconda installer init version 13.21.239 starting
mounting /proc filesystem... done
creating /dev filesystem... done
starting udev...done
mounting /dev/pts (unix98 pty) filesystem... done
```

```
mounting /sys filesystem... done
anaconda installer init version 13.21.239 using a serial console
trying to remount root filesystem read write... done
mounting /tmp as tmpfs... done
running install...
running /sbin/loader
Gdetecting hardware...
waiting for hardware to initialize...
detecting hardware...
waiting for hardware to initialize...
Looking for installation images on CD device /dev/
sr0Running anaconda 13.21.239, the SCL system installer - please wait.
Examining storage devices
In progress

Creating device /dev/sda1
Creating ext3 filesystem on /dev/sda1
In progress

Creating device /dev/sda2
Creating swap on /dev/sda2
In progress

Creating device /dev/sda3
Creating ext3 filesystem on /dev/sda3
In progress

Retrieving installation information for SCL.
In progress

Retrieving installation information for SCL.
Checking dependencies in packages selected for installation
In progress
Completed

Starting installation process
In progress

Preparing transaction from installation source
In progress

Installing libgcc-4.4.7-16.SCLC6_6.R2.0.1.x86_64 (520 KB)
GCC version 4.4 shared support library
Packages completed: 1 of 646
Installing setup-2.8.14-20.SCLC6_6.1.R1.0.1.noarch (650 KB)
A set of system configuration and setup files
/* ...      Output Truncated for brevity
... */
Packages completed: 646 of 646
Performing post-installation configuration
Installing bootloader.
Running post-installation scripts
Running poweroff...
```

Obtaining a License

Once the software is installed, a license file is needed in order to start the Path Computation Server (PCS) services. Instructions on how to obtain the license file are emailed with product purchase.

1. After the server is built, log in as the root user with the default credentials (password: *northstar*).
2. Take note of the interface and MAC address of the server (either eth0 or eth1 will do).

```
[root@pcs ~]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr AA:BB:CC:DD:EE:FF
          inet addr:1.2.3.4  Bcast:  Mask:255.255.255.0
          inet6 addr: fe80::5468:aaff:fedd:eeff/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

3. Send an email to NORTHSTAR-KEY-REQUEST@juniper.net with the following information:

- In the Subject header: *NorthStar Product Keys Needed*
- Ethernet card type and MAC: In our example these would be eth0 with MAC AA:BB:CC:DD:EE:FE.
- Company and contact info: Including company name, primary contact name, full company address, contact email address contact phone number
- Number of nodes purchased
- Finally, include the purchase order info so it will be possible to verify the purchase.

A product key will be sent back, which consists of a text file that needs to be copied to the server under the */opt/pcs/db/sys* folder. The name of the file must be *npatpw* and should be readable by the PCS user.

Once the license is uploaded, the PCS service needs to be started. You can simply reboot the server (*/sbin/reboot*) and the startup process takes care of bringing up all the required services and daemons. Alternatively, when running version 2.0 or higher, you can manually start up the PCS processes by using the *supervisorctl* command, which takes care of launching and monitoring the different processes:

```
[root@northstar ~]# supervisorctl restart all
infra:keepalived: stopped
listener1:listener1_00: stopped
junos:junosvm: stopped
northstar:npat_ro: stopped
```

```

northstar:pcserver: stopped
northstar:toposerver: stopped
northstar:pcserver: stopped
northstar:mladapter: stopped
northstar:npat: stopped
infra:haproxy: stopped
infra:nodejs: stopped
infra:zookeeper: stopped
infra:rabbitmq: stopped
infra:ha_agent: stopped
infra:cassandra: stopped
listener1:listener1_00: started
northstar:mladapter: stopped
junos:junosvm: started
northstar:npat_ro: started
northstar:pcserver: started
northstar:toposerver: started
northstar:pcserver: started
northstar:npat: started
infra:haproxy: started
infra:nodejs: started
infra:rabbitmq: started
infra:ha_agent: started
infra:zookeeper: started
infra:keepalived: started
infra:cassandra: started

```

CAUTION If you try to start the server without a license or with an invalid license file, the process will fail to start. The log file for the process (located in */opt/pcs/log/pcs.log*) will indicate that the license is missing.

The status of the different processes is reported by the supervisor daemon, which can also be used to restart individual processes:

```

[root@northstar ~]# supervisorctl status
infra:cassandra                RUNNING pid 6060, uptime 0:08:08
infra:ha_agent                 RUNNING pid 6055, uptime 0:08:08
infra:haproxy                  RUNNING pid 5858, uptime 0:08:10
infra:keepalived               RUNNING pid 5878, uptime 0:08:09
infra:nodejs                   RUNNING pid 5860, uptime 0:08:09
infra:rabbitmq                 RUNNING pid 5881, uptime 0:08:08
infra:zookeeper                RUNNING pid 5865, uptime 0:08:09
junos:junosvm                  RUNNING pid 5761, uptime 0:08:12
listener1:listener1_00        RUNNING pid 5760, uptime 0:08:12
northstar:mladapter            RUNNING pid 6362, uptime 0:07:58
northstar:npat                 RUNNING pid 5844, uptime 0:08:10
northstar:npat_ro              RUNNING pid 5802, uptime 0:08:12
northstar:pcserver             RUNNING pid 5815, uptime 0:08:12
northstar:pcserver             RUNNING pid 5838, uptime 0:08:11
northstar:toposerver           RUNNING pid 5822, uptime 0:08:12

```

Connecting the Controller to the Network

With the server installed, you now need to connect it to the network. The default networking configuration allows both the use of a single connection between the controller and the network, or the use of separate networks, one for connecting to the routers, and one used to access the GUI (normally connected to the management network).

Figure 1.3 shows the default network connectivity and addresses configured in a default install

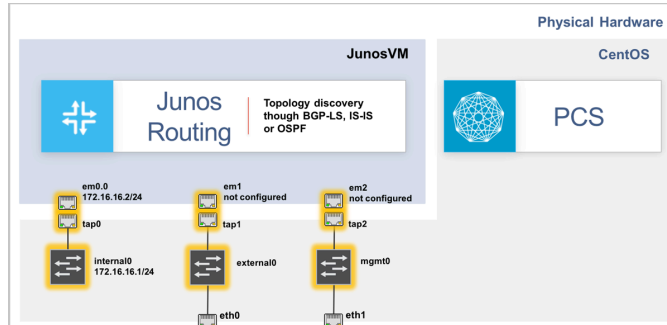


Figure 1.3 Default Network Configuration of the Controller

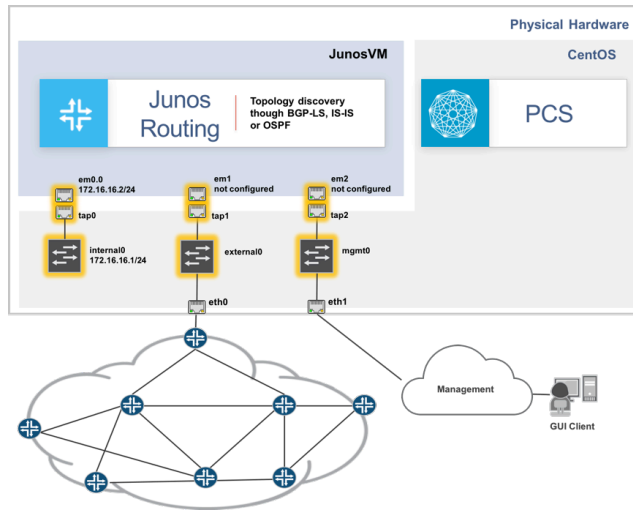


Figure 1.4 External and Management Networks

A virtual machine (VM) running the Junos OS is included with the controller and used for discovering the network topology (more on that later). This VM is bundled with the controller so you don't have to worry about installing it, but you need to keep in mind that it's there when you connect the controller to the network.

Typically, the external0 bridge connecting both the eth0 interface in the host OS and the em1 interface in the JunosVM is used to connect to the routers. The GUI clients and other management stations are normally connected to the eth1 interface in the host which, in turn, is connected to the em2 interface in the JunosVM through the mgmt0 bridge. The use of the eth1 interface (and consequently the em2 interface in the JunosVM) is completely optional – the GUI can be accessed both from the eth0 and eth1 interfaces.

Changing the interface addresses and basic network connectivity settings, like adding or removing static routes from both the JunosVM and the CentOS host image, can be easily done with the `net_setup` script, located under the `/opt/northstar/util` folder:

```
[root@northstar ~]# /opt/northstar/utis/net_setup.py
```

Main Menu:

```
.....
A.) Host configuration
B.) JunosVM configuration
.....
C.) Check Network Setting
.....
D.) Maintenance & Troubleshooting
.....
E.) HA Setting
.....
F.) Collect Trace/Log
.....
X.) Exit
.....
```

Please select a letter to execute.

Let's connect the controller to the network as shown below:

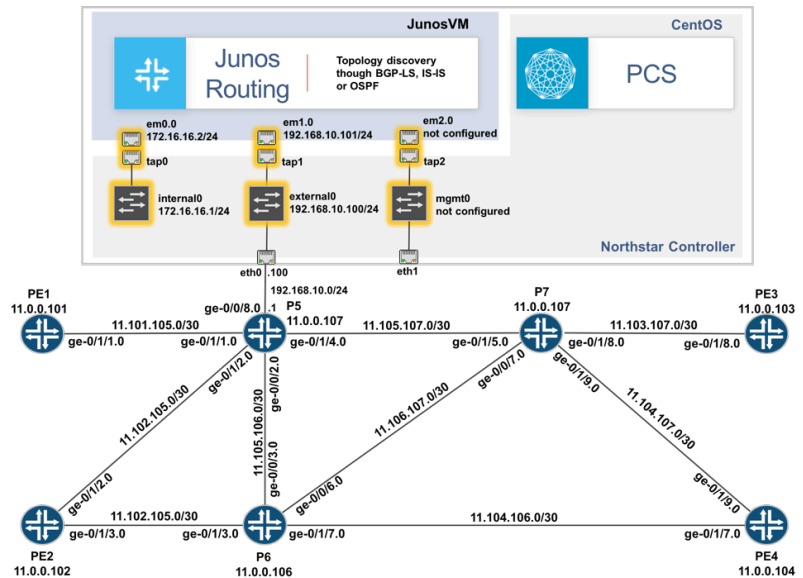


Figure 1.5 Connecting the Controller to the Network

1. In the controller, configure the following IPs for both the host and Junos VM:

- Host external/router-facing IP: 192.168.10.100
- Host external netmask: 255.255.255.0
- Host default gateway: 192.168.10.1
- JunosVM external IP: 192.168.10.101
- JunosVM external netmask: 255.255.255.0

Host Configuration:

```

.....
1.) Host external/router-facing IP           : 192.168.10.100
2.) Host external/router-facing Netmask      : 255.255.255.0
3.) Host mgmt/user-facing IP (optional)      : 0.0.0.0
4.) Host mgmt/user-facing Netmask (optional) : 0.0.0.0
5.) Host default gateway                     : 192.168.10.1
6.) Show host current static route
7.) Show host candidate static route
8.) Add Host candidate static route
9.) Remove Host candidate static route
.....
A.) Apply Host static route only
B.) Apply Host Setting
.....

```

Junos VM Configuration Settings:

```

.....
1.) JunosVM External/router-facing IP       : 192.168.10.101
2.) JunosVM External/router-facing Netmask   : 255.255.255.0
3.) JunosVM mgmt/user-facing IP (optional)   : 0.0.0.0
4.) JunosVM mgmt/user-facing Netmask (optional) : 0.0.0.0
5.) JunosVM default gateway                 : 192.168.10.1
6.) Show Junos VM current static route
7.) Show Junos VM candidate static route
8.) Add Junos VM candidate static route
9.) Remove Junos VM candidate static route
A.) BGP AS Number                          : 100
.....
B.) Apply JunosVM static route only
C.) Apply JunosVM Setting
.....

```

2. On the P5 router, which connects directly to the controller, you need to configure interface ge-0/0/8:

```

[edit interfaces]
root@P5# show
/* ... other interfaces were omitted ... */
ge-0/0/8 {
    unit 0 {
        family inet {
            address 192.168.10.1/24;
        }
    }
}

```

3. Once you connect P5 to the controller though the eth0 interface, make sure you can ping both the host and the JunosVM from P5, or whatever network connection you've used for the controller in your own lab.

Topology Discovery

Okay, now that there's a working test bed, the first order of business is to use the controller to discover the topology. The controller uses the topology information to find out the different paths available in the network to route LSPs.

Clearly, the network is already aware of the overall topology since this information is used by each node in order to signal new LSPs. So instead of trying to build the topology from either the configuration files or from topology-discovery protocols, the management station can participate passively in the IGP. After all, link-state database routing protocols (such as OSPF or IS-IS) go through great lengths to make sure the topology of the network is known to the different nodes within an area, while making sure changes in the database are promptly synchronized across all devices in the area. So, instead of trying to discover the topology through out-of-band mechanisms, the NorthStar Controller uses a Junos VM that participates in the IGP and builds a representation of the network's topology from its perspective, as shown in Figure 1.6. This representation is kept up-to-date and will adjust to changes in the network as soon as the IGP propagates the changes.

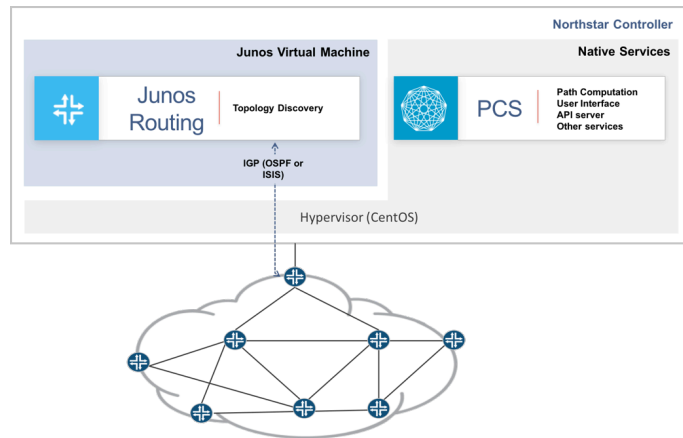


Figure 1.6 Topology Discovery Using an IGP

While this has many of the desired properties for a topology acquisition mechanism, careful engineers will note that by following this approach, you also inherit some of the limitations of the IGP's used to propagate this information. In particular:

- In order to scale the IGP, large networks are partitioned into multiple areas. Topology (and traffic engineering) information is only propagated within an area, and routing between areas is done by sending traffic to border devices (devices sitting between

areas). Instead of having full knowledge of the topology of the full network, devices only know the topology of the areas they belong to, while receiving some abstract data representing the different prefixes that are reachable through other areas (and which devices can be used to bridge the areas). It is obvious, then, that in large networks it isn't enough to simply participate in the IGP; you also need a mechanism to collect data from *all the* different areas (and even autonomous systems).

- Most IGPs require a direct connection between the different nodes (after all, part of their job is to find out the different paths in the network), yet sometimes out-of-band controllers like NorthStar tend to be installed in management networks with no direct connection to the devices.
- The controller connects to the network only to obtain information from it, no data traffic should ever be sent to the controller. However, if the controller is connected to the network with redundant paths (desirable in order to minimize single points of failure), special care must be taken to guarantee that other nodes in the network don't choose the controller when calculating the best paths.

The solution? Put the controller *behind* one or more routing elements that do not participate in the IGP. And this is accomplished by configuring a GRE tunnel with routing enabled over the tunnel, effectively jumping over the intermediate nodes and simulating a direct connection between the controller and the rest of the network, as shown in Figure 1.7.

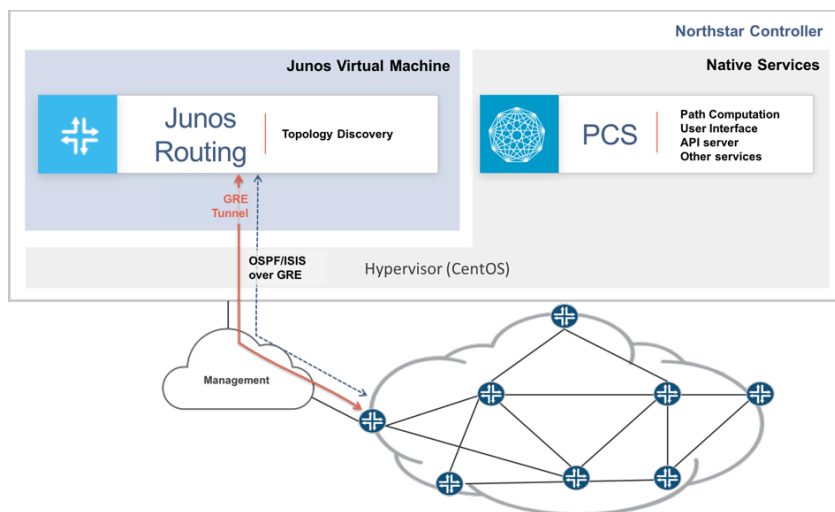


Figure 1.7 Using GRE to Bypass Intermediate Nodes

But another, better alternative, exists that does not risk having the controller in the path of the traffic. All those IGP issues can be solved if you realize that all the controller needs is a way to extract the traffic engineering database (TED) that the nodes build.

Within each area, the TED is the same in all nodes (except for some transient changes, like utilization changes, that are only synchronized periodically), so a different approach can be taken – the controller connects to the nodes in the different areas (or autonomous systems). Using the powerful BGP, extensions have been developed that allow devices to export the contents of the TED using BGP (using draft-ietf-idr-ls-distribution-11, or more commonly referred to as BGP-LS, for *link-state* distribution). Figure 1.8 represents this solution.

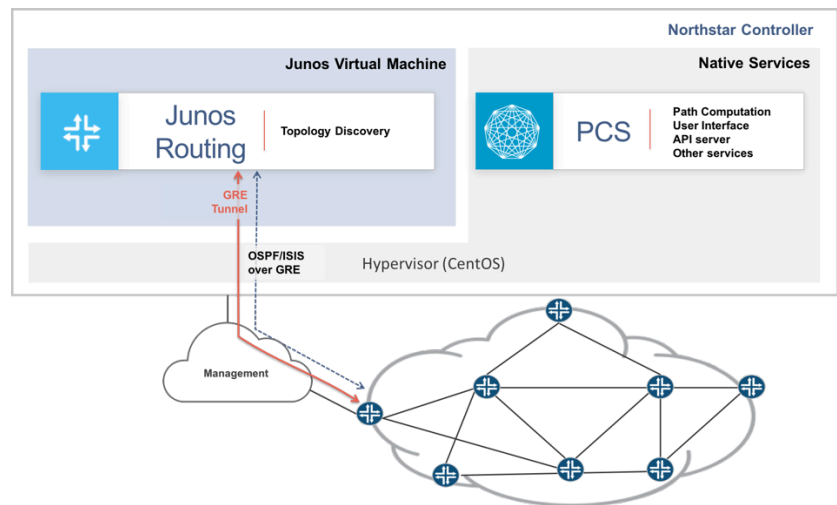


Figure 1.8 Using BGP-LS to Extract Topology Information

Don't worry, you'll get back to the lab, but first a little more on BGP-LS and why you're going to want to use it.

BGP-LS in Multi-Area Topologies

Multi-area topologies can be gracefully handled when using BGP-LS since the TED is synchronized across all devices within an area, and you only need to peer with one device in each area, as shown in Figure 1.9.

The configuration is identical to the single-area BGP-LS case, but you are peering with at least one device in each area and that makes border routers a natural choice (refer to the configuration in this chapter's section, *Connecting the Controller to the Network*). Let's leave it at

that right now, since it's probably better to use the topology in the GUI to visualize the changes once you start changing the OSPF configuration.

Anyway, the next few sections explore how to configure the network and NorthStar controller.

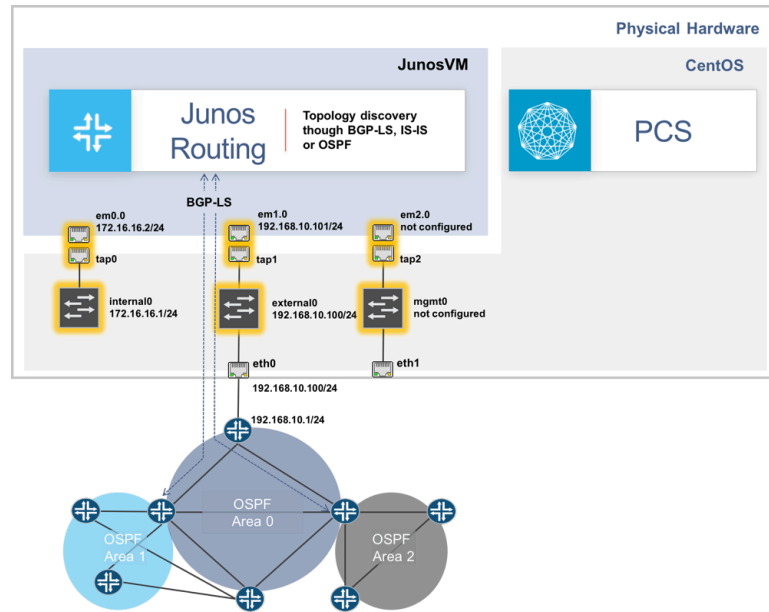


Figure 1.9 Multi-Area Topologies

Using BGP-LS

This *Day One* book uses the most common deployment that is not susceptible to the before-mentioned IGP limitations. It's also the simplest to configure and so will be the starting point to get the system up and running. Later the book will show you how to configure topology discovery *using an IGP* for those of you that might like to experiment.

By default, the Junos VM is configured to accept BGP connections from any device in the network using a local AS of 100. The local AS number can be changed by using the `net_setup.py` script. There is nothing to configure in the Junos VM if BGP-LS is going to be used (apart from, maybe, changing the AS number used by BGP). But, you do need to configure one of the devices in the network to peer with the controller and export all the contents of the TED using BGP.

As with other address families, the Junos OS stores the information in per-address family tables (inet.0, mpls.0, iso.0, etc.). For the traffic-engineering family, the data is stored in the *lsdist.0* table, so you need a routing policy to leak the contents of the TED into the *lsdist.0* table and then advertise these routes using BGP-LS. The topology is illustrated in Figure 1.10.

We'll configure a BGP peering session between the JunosVM and P6, using the loopback IP in P6 to terminate the session.

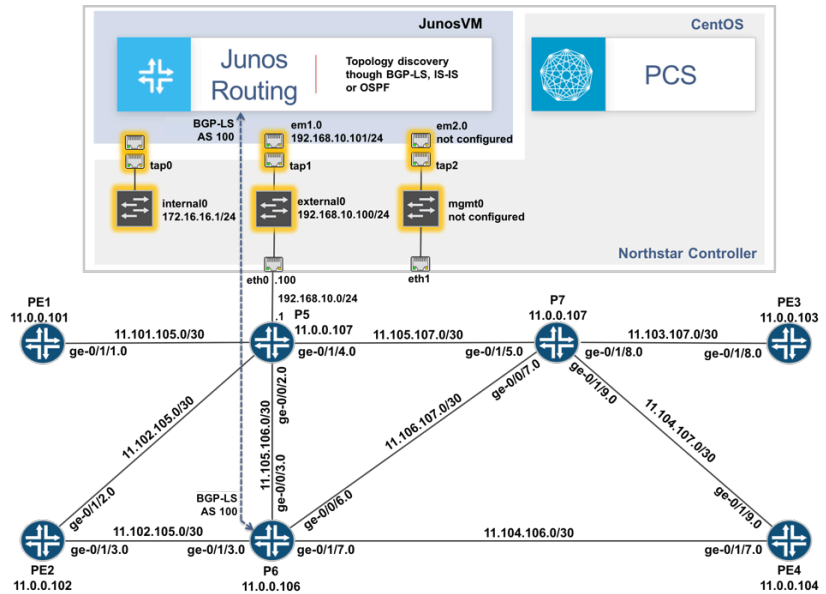


Figure 1.10 BGP-LS Topology

The configuration required in the P6 used to export the traffic engineering database is shown below. The IGP (either OSPF or IS-IS) is already configured in the network and traffic-engineering extensions are in use to propagate the information required by RSVP to compute MPLS paths:

```
[edit protocols bgp]
root@P6# show
group northstar {
    type internal;
    /* This is the IP of the interface in the local router used to connect to the controller. A
    loopback can be used, and in both cases make sure you have all the required static routes
    configured in the controller to reach this IP. */
    local-address 11.0.0.106;
    family traffic-engineering {
        unicast;
    }
}
/* By default BGP does not export any route from the lsdist.0 table, so you need to add an
export policy to advertise these routes to the controller */
```

```

    export TE;
/* #This is the IP address configured in the JunosVM */
    neighbor 192.168.10.101;
}
[edit protocols mpls traffic-engineering]
root@P6# show
database {
    import {
/* This is required so that the contents of the TED are injected into the lsdist.0 table */
        policy TE;
    }
}
[edit policy-options]
root@P6# show
policy-statement TE {
    from family traffic-engineering;
    then accept;
}
[edit routing-options]
root@P6# show
autonomous-system 100;

```

And, in case you are curious, the BGP configuration of the Junos VM is shown below. Note the use of the `allow 0.0.0.0/0` command to enable any device from any network to establish a BGP session to the controller:

```

[edit protocols bgp]
root@northstar_junosvm# show
group northstar {
    type internal;
    family traffic-engineering {
        unicast;
    }
    allow 0.0.0.0/0;
}

```

Summary

You should now have all the information on how to build a controller, how to connect it, and what to configure in your lab, whether you are simulating this book's topology or not.

Make sure everything is working because it's time to fire up the NorthStar Controller and show off what you have just done!

Chapter 2

Using the NorthStar Controller

Once the topology is discovered, the nodes and links connecting them will be automatically visible in the network topology.

Using the NorthStar GUI

You can access the NorthStar graphic user interface (GUI) either through the eth0 or eth1 interfaces. Connect to the controller with an http to <http://192.168.10.100:8091> or an https to <https://192.168.10.100:8443> and you'll get the landing page of Figure 2.1. If you are connecting to the controller using the eth1 interface (assuming the eth0 interface is used to connect the controller to the routers), you'll need to use the *net_setup.py* script to set up the Internet Protocol (IP) of the eth1 and em1 interfaces.

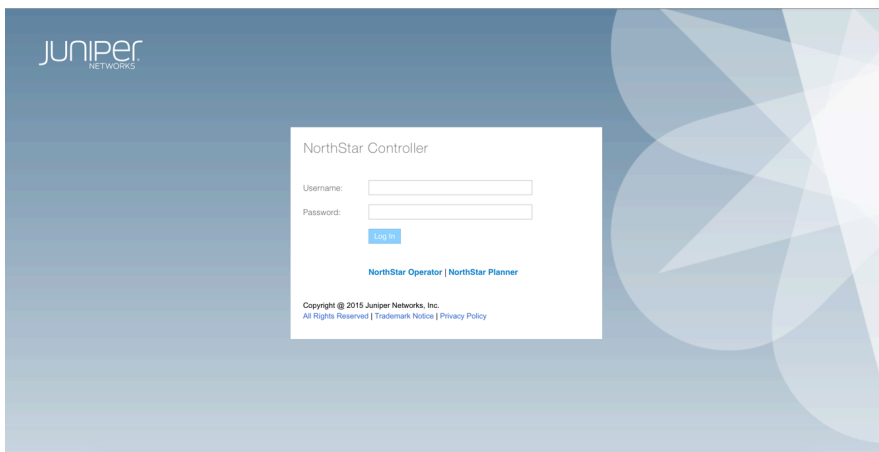


Figure 2.1 NorthStar 2.0 Landing Page

Log in with the default credentials, *admin/admin*. Click on the Topology tab in the menu bar and the discovered network topology will be visible as shown in Figure 2.2.

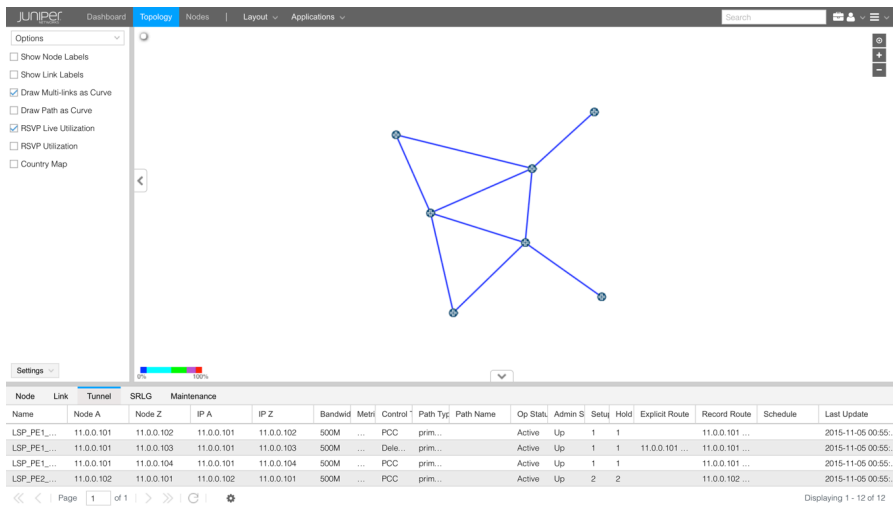


Figure 2.2 Initial Topology

There are a few things to keep in mind when viewing these topology visualizations:

- The layout can be set arbitrarily, or it can be automatically computed. Depending on the size of the network and the aspect of the network you are inspecting, different layouts may prove to be more suitable than others. Users can either use one of the auto layout functionalities (right click in the map window and select a layout menu item) or manually move the nodes to their desired position. Layouts can be saved by right clicking the Map View window and selecting the Map Views menu item. (Previously saved layouts can be loaded from this same window.)
- In particular, the location of a node can be used to build a geographical layout. Since this information cannot be automatically discovered from the traffic engineering database, it is necessary to enter the location of each node manually if you want to show the geographical topology. Select a node in the Nodes table at the bottom of the screen. Click the Modify button and enter the location (Latitude/Longitude). You can also set the background image to display a map, by clicking the Country Map checkbox in the options panel. Figure 2.3 shows an example of such view.

- It is possible to add additional information that can't be discovered from the traffic engineering database, such as link delays (REST APIs can be used to automate this process), admin weights, node names, etc. All of it can be used by the controller when calculating paths.
- The UI provides several ways to filter the nodes and links displayed based on a number of criteria such as the protocols they have enabled, their AS type, etc.

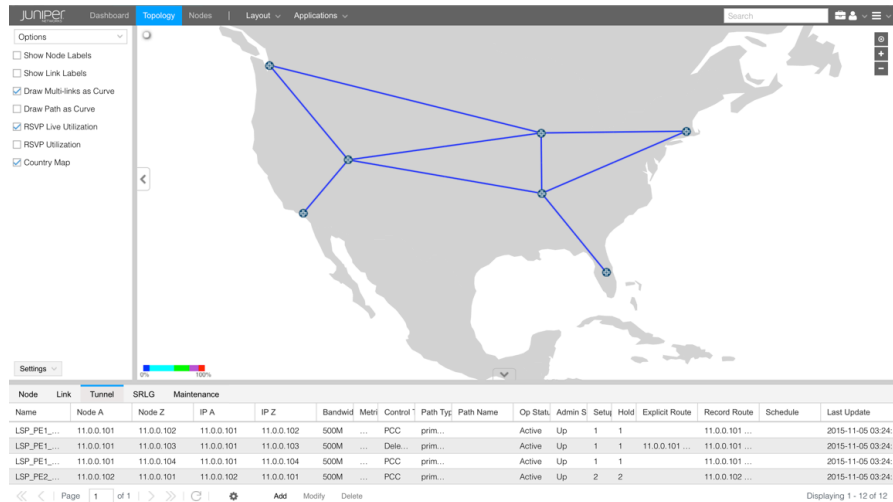


Figure 2.3 Example Geographical Topology

The GUI provides a lot of flexibility when dealing with the different ways to visualize the network, group and arrange the nodes in the canvas, and monitor the status of the links and nodes. It's not the objective of this *Day One* book to explore the NorthStar Controller GUI (that's the focus of other Getting Started documents), but take the time to explore the GUI and its many features.

For example, a lot of information is obtained by the controller that can be displayed in the topology to help visualize different aspects of the network, as shown in Figure 2.4.

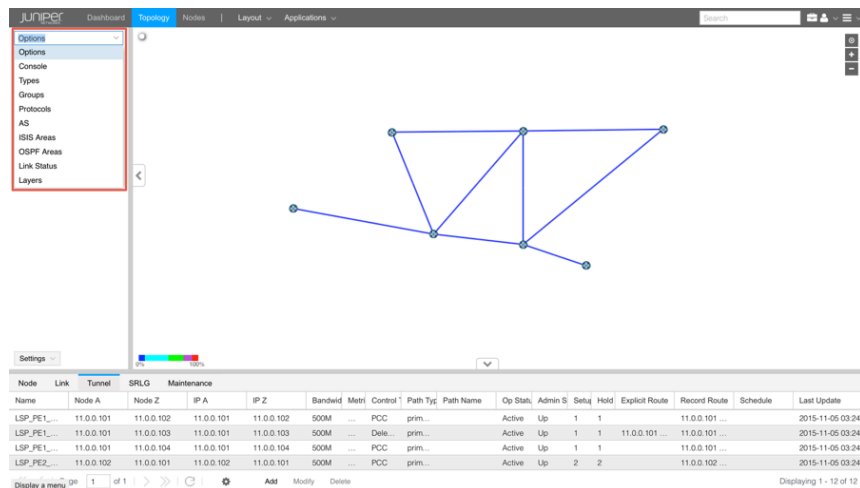


Figure 2.4 Selecting Views

The options are accessed from the top left corner of the Map window and include some of the following views and sub-views:

- **RSVP Utilization:** This view colors the links according to their planned utilization. The links will account for the bandwidth reserved by all the tunnels routed through them, even if they haven't yet been pushed to the network (i.e., they are in a *planned* state).
- **RSVP Live Utilization:** Similar to the previous view, but this view only accounts for the capacity utilized by the tunnels provisioned in the network.
- **AS Subview:** This view colors the nodes based on the AS they belong to.
- **Coloring Subview:** This view shows or hides all the links that belong to one or more admin groups (java client only).
- **OSPF Area:** This view colors links according to the OSPF area they belong to.
- **IS-IS Area:** This colors the nodes according to the ISIS area they belong to.
- **Protocols:** This shows or hides links based on the protocols they are running.
- **Types:** This view shows the different types of nodes and allows it to filter the nodes by type. The Java client UI (accessible from the login page by clicking the NorthStar operator link) also colors

links based on their type, and can be used to hide certain types of links, when you wish.

As you work on more complex topologies you'll find some of these views quite useful, like running mosaics all at once as shown in Figure 2.5. Oftentimes you can spot issues with a simple glance – you'll remember the good old days when you had to crawl through the device racks trying to figure out why some LSP wasn't coming up. Give it a try. Disable RSVP from an interface and watch what happens!

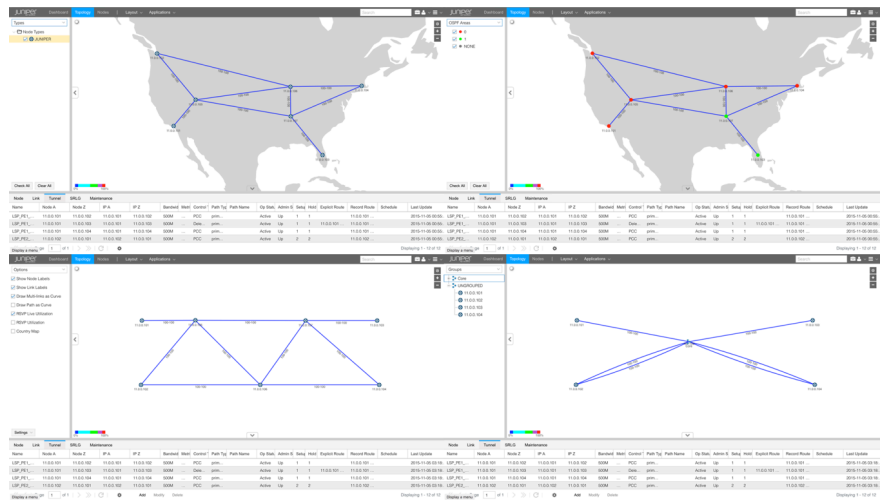


Figure 2.5 View Mosaic

A cool feature of the topology map to explore is NorthStar's ability to label nodes and links using the data discovered through PCEP and the information from the TED. Select Options from the drop down menu (see Figure 2.2) and select the Show Node Labels and Show Link Labels checkboxes. At the bottom of the Options panel is a Settings button where you can change what labels are used for your links and nodes.

NOTE At the time of this writing, the Java client can use more information to label links and nodes.

Figure 2.6 adds a few labels just to give you an idea of how NorthStar visualizes the information stored in the topology database. Some of this information is derived from data that has to be configured manually, or through the REST APIs, such as the geographic distance of a link based on the location of the endpoints. And some of the labels that can be displayed include information that is not yet available to the controller, but put there as placeholders for new discovery mechanisms as they become integrated into the product.

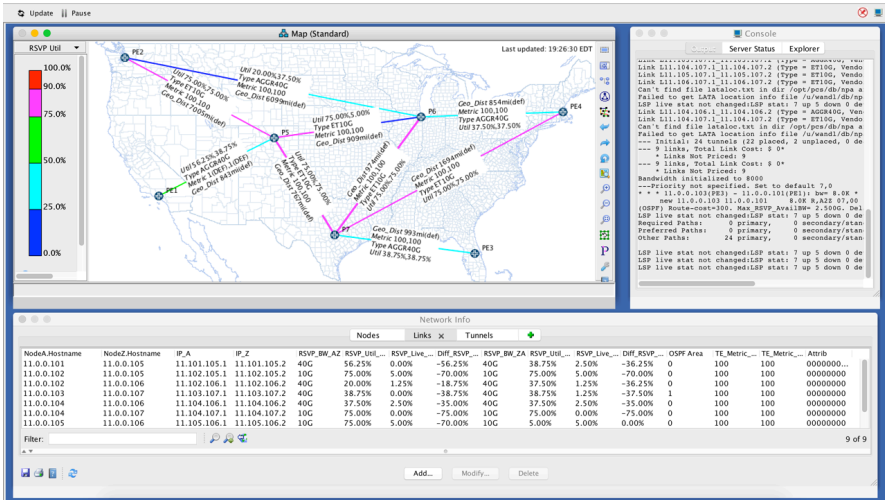


Figure 2.6 Customizing Node and Link Labels

Representing Network Types

You are probably already familiar with the concept of *pseudo nodes* in OSPF and IS-IS. But if you haven't touched these standards in a long time, a quick reminder is in order.

OSPF and IS-IS deal with broadcast or multi-access networks by forming adjacencies with the designated router (DR), or designated intermediate system (DIS) in the case of IS-IS. In both cases, they represent the connectivity between all nodes in the LAN segment by generating a pseudo node. In the topology model representation, all nodes in the network connect to the pseudo node so all nodes in the network segment can send traffic to each other by going through the pseudo node. Representing the topology in this manner reduces the number of adjacencies and state (size of the topology database) that nodes in a multi-access network need to maintain.

So, when showing these kinds of networks, the controller displays the pseudo node in the topology with all nodes in the same network segment connecting to it (in other words, the controller simply displays the network topology in much the same way as it is represented in the IGP's database).

If, for example, you change the OSPF interface type to their default types (LAN) on both endpoints of a link, you'll see a pseudo node show up in the topology. Let's do just that and change the OSPF interface type of the interfaces connecting PE1 and P5 (ge-0/1/1.0 in both devices) back to their default type:

```
[edit protocols ospf]
root@PE1# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.0 {
    interface fxp0.0 {
        disable;
    }
    interface all {
        interface-type p2p;
    }
    interface ge-0/1/1.0;
}
```

And in P5 as well:

```
[edit protocols ospf]
root@P5# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.0 {
    interface all {
        interface-type p2p;
    }
    interface fxp0.0 {
        disable;
    }
    interface ge-0/1/1.0;
}
```

And now the pseudo node is being shown (see Figure 2.7) in the NorthStar GUI connecting the PE1 and P5 nodes.

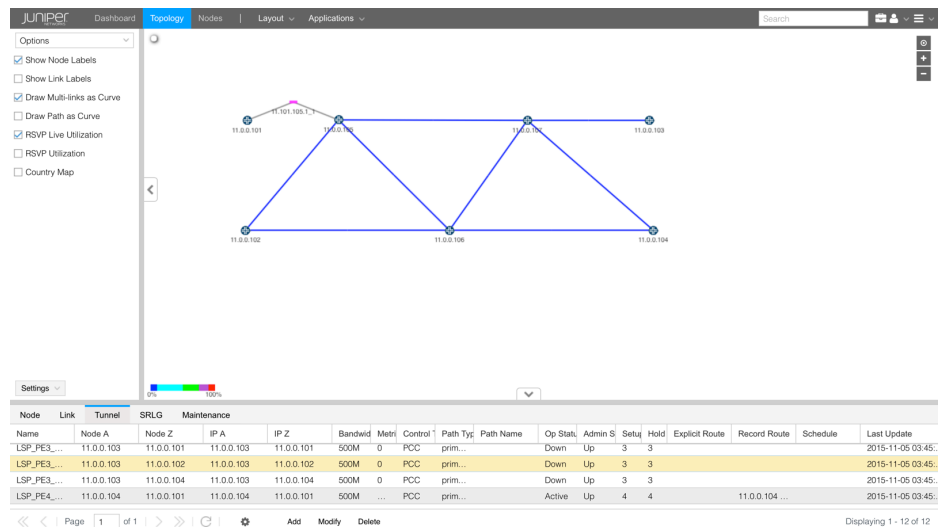


Figure 2.7 Pseudo Nodes

Because some protection mechanisms work better over P2P links, and there isn't any reason not to configure Ethernet links as P2P in the transport, expect all links in the topology to be shown as simple point-to-point lines. And here the presence of pseudo nodes can alert you to possible misconfigurations.

Multi-Area Topologies

Now that you can view the topology graphically in your lab, let's change the OSPF configuration so that node PE3 is in a different OSPF area. The resulting topology should end up being something like Figure 2.8.

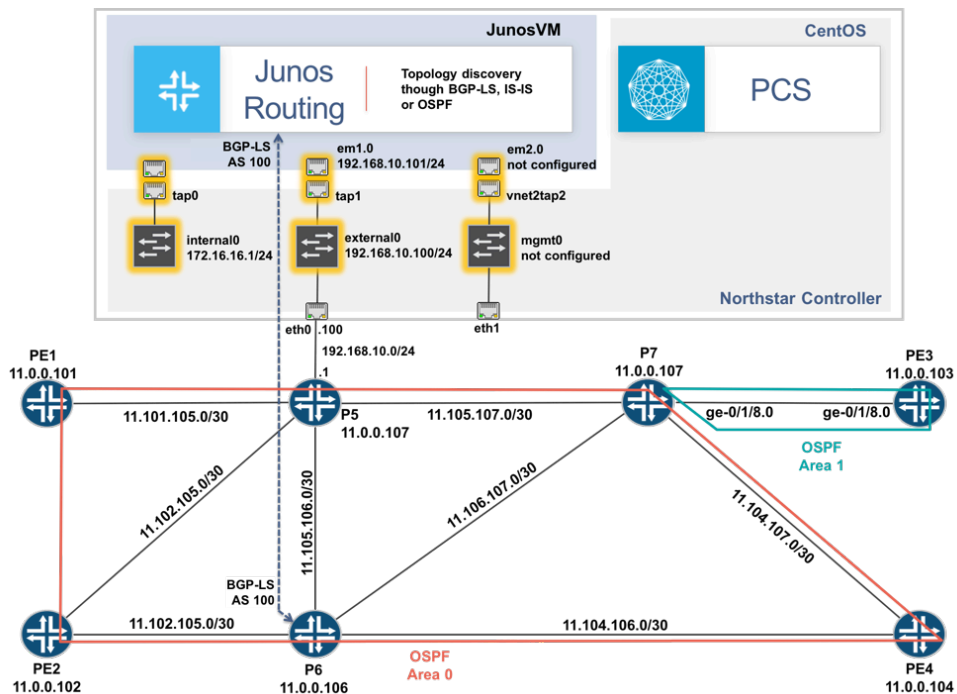


Figure 2.8 Dual Area Topology

IMPORTANT

Just to make things clear, you can arrange the nodes in the NorthStar GUI to match the layout in our diagram or in any way you find convenient to your and your lab. Like Figure 2.9.

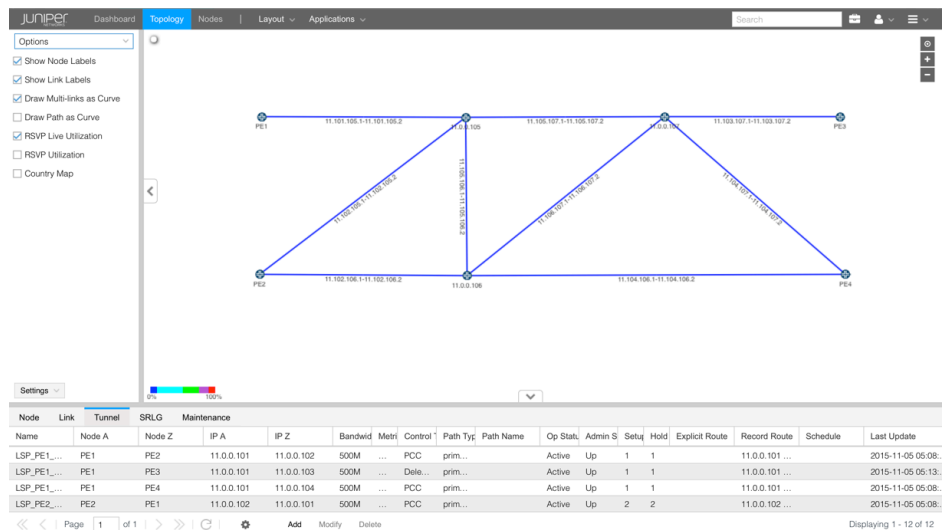


Figure 2.9 Logical Topology

To change the OSPF area in P3 and add area 1 in P7 with the ge-0/1/8.0 interface in it, the OSPF configuration in P7 looks like this:

```
[edit protocols ospf]
root@P7# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.0 {
    interface all {
        interface-type p2p;
    }
}
area 0.0.0.1 {
    interface ge-0/1/8.0 {
        interface-type p2p;
    }
}
```

And in PE3, the resulting configuration is:

```
[edit protocols ospf]
root@PE3# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.1 {
    interface all {
        interface-type p2p;
    }
    interface fxp0s.0 {
        disable;
    }
}
```

Once changed, the OSPF adjacency should be rebuilt:

```
root@PE3# run show ospf neighbor
Address      Interface    State    ID          Pri    Dead
11.103.107.2 ge-0/1/8.0  Full    11.0.0.107  128    36
```

Yet, if you look at the NorthStar's GUI in Figure 2.10 you'll see that it looks like the P7 to PE3 connection is down!

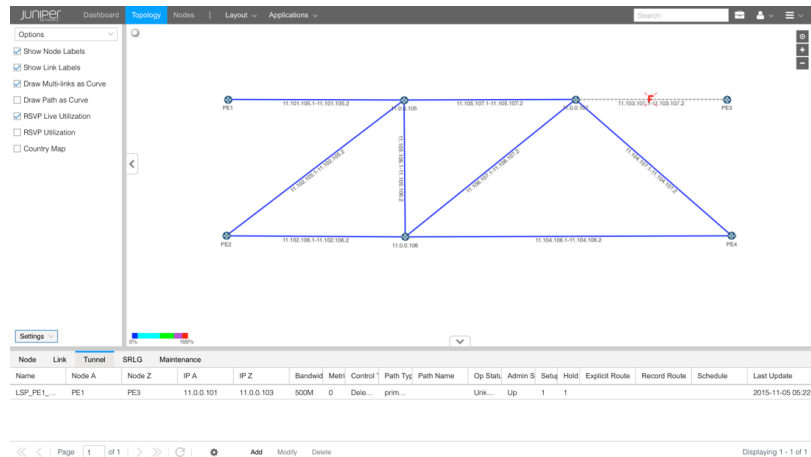


Figure 2.10 Topology View After OSPF Area Change

Why can't the controller see node PE3? Obviously (after all those discussions regarding multi-area topologies), what is happening is that P6 has no view of what's going on in the OSPF area 1, and NorthStar is extracting the topology from P6 alone. We chose to change PE3 because we didn't have to make many changes in the network but we also made sure that P6 did not become a border router for this new area. If P6 were a border router, we wouldn't need to make any changes to the topology discovery since the NorthStar controller would get all the information it needed from P6 alone.

So P7 has become a border router of area 1 and that makes it a good candidate for extracting the TED through BGP-LS. The additional configuration on P7 is identical to the one already done on P6 (apart from the local address used to establish the BGP session), as you can see from the configuration snippet below:

```
[edit protocols bgp]
root@P7# show
group Northstar {
  type internal;
  local-address 11.0.0.107;
  family traffic-engineering {
    unicast;
  }
}
```

```

export TE;
local-as 100;
neighbor 192.168.10.101;
}
[edit protocols mpls traffic-engineering]
root@P7# show
database {
    import {
        policy TE;
    }
}
[edit policy-options]
root@P7# show
policy-statement TE {
    from family traffic-engineering;
    then accept;
}

```

Now you could choose to extract the topology *only* from P7 since it does connect to all the areas in the domain, but for redundancy purposes it is still a good idea to export the topology from a couple devices. Large topologies with many areas are unlikely to have a single device acting as a border router of all the areas, so in most cases, peering with several devices will still be required.

And the fun part is now visualizing the changes from the NorthStar Controller's GUI. From the main Topology tab window, select the OSPF area sub view. It will color the nodes and interfaces according to the areas they belong to, like the screen capture in Figure 2.11.

We've found in the lab that it's useful to see how the controller obtained this information, and it's a good exercise to poke around the TED and the lsdist.0 tables in the different nodes to get an idea of the information the controller receives.

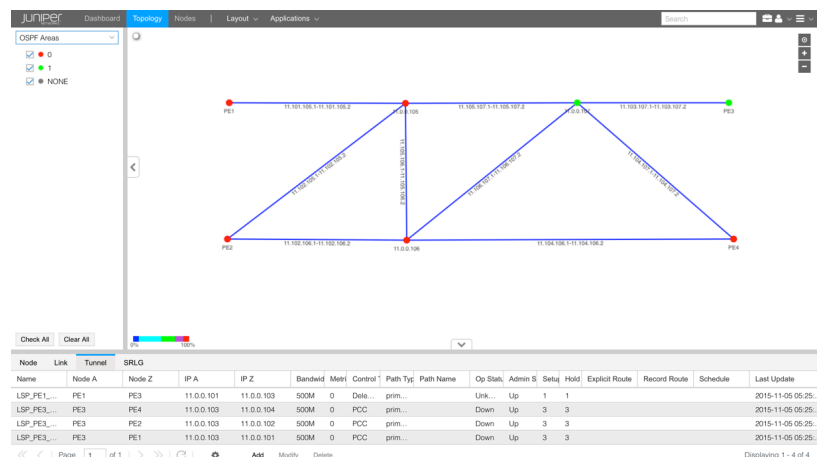


Figure 2.11 Multi-Area Topology View

Using OSPF or IS-IS (Optional)

Most networks use BGP-LS to obtain topology information, because it is simple and straightforward, but it's still instructive to view how to learn the TED directly from an IGP. Yet there is also an advantage of using an IGP over using BGP-LS to obtain the topology information. Since BGP-LS only carries the contents of the TED, links that are not eligible to carry MPLS traffic are not included in the topology, such as links without MPLS or RSVP enabled. By using an IGP it is possible to learn the full topology, which can be useful for monitoring purposes. Let's start with the baseline network shown previously.

The NorthStar Controller is connected to the network through P5, and in P5, all interfaces have OSPF enabled in area 0. All you have to do is configure OSPF in the JunosVM so the traffic engineering database can be populated.

There are a few ways to access the JunosVM:

- If the JunosVM IP is configured, you can use ssh (in this example, you would ssh with the root user to the 192.168.10.101 IP) to the JunosVM to configure the routing protocol.
- When the JunosVM is instantiated, a console connection listening on port 50002 is always brought up. So, even without any IPs configured in the JunosVM, it is always possible to reach its console by doing a telnet to port 50002 on any of the host OS addresses (including the loopback):

```
[root@northstar ~]# telnet 127.0.0.1 50002
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^J'.

login: root
Password:
--- JUNOS 14.2X1.1 built 2015-04-11 01:49:40 UTC
root@northstar_junosvm% cli
root@northstar_junosvm>
```

Log in to the JunosVM and configure OSPF as you would normally do for other devices in the network. In this book's example, you have all the routers in area 0 and you'll enable OSPF in the interface em1.0, which connects to P5:

```
[edit Protocols ospf]
root@northstar_junosvm# show
traffic-engineering;
area 0.0.0.0 {
    interface em1.0 {
        interface-type p2p;
    }
}
```

```

interface lo0.0 {
    passive;
}

```

MORE? Detailed examples of how to configure OSPF can be found in the Juniper TechLibrary’s *OSPF Configuration Guide*: http://www.juniper.net/techpubs/en_US/junos14.2/information-products/pathway-pages/config-guide-routing/config-guide-ospf.html.

When you are connected to the JunosVM, you might as well check the configuration of the em1 interface to make sure that the script correctly pushed the interface configuration (you can always make any required changes to the JunosVM directly from the Junos CLI):

```

[edit interfaces]
root@northstar_junosvm# show
em0 {
    unit 0 {
        family inet {
            address 172.16.16.2/24;
        }
    }
}
em1 {
    unit 0 {
        family inet {
            address 192.168.10.101/24;
        }
    }
}

```

For completeness, a similar IS-IS configuration is shown below. We have excluded the configuration of the ISO address on the loopback interface and “family iso” in the em1.0 interface for brevity:

```

[edit protocols isis]
root@northstar_junosvm# show
traffic-engineering igp-topology;
level 1 disable;
level 2 wide-metrics-only;
interface em1.0 {
    point-to-point;
}
interface lo0.0;

```

MORE? Similar to OSPF, you can find detailed examples of how to configure ISIS in the Juniper TechLibrary’s *ISIS Feature Guide*: http://www.juniper.net/techpubs/en_US/junos14.2/information-products/pathway-pages/config-guide-routing/config-guide-routing-is-is.html.

NOTE One advantage of using IS-IS over OSPF is that the hostnames of the devices participating in the IGP will be automatically discovered and available to the controller, as per [RFC 5301](#).

The topology export process exports all the traffic engineering database not from the TED, but rather the `lsdist.0` table. This may seem redundant, but remember that this is the table used to store traffic engineering information injected into BGP and that's where the topology export process, running the Junos VM, picks up the data for the NorthStar Controller.

When you use BGP-LS, you have to inject the topology information into the `lsdist.0` table so that BGP could advertise the data to its peers. The situation here is analogous as when you learn the TED from the IGP and leak the data into the `lsdist.0` table, but instead of doing this in the advertising router, its done in the JunosVM.

Once the data from the TED is injected into the `lsdist.0` table, the topology model will be automatically constructed. Let's add the missing configuration to the JunosVM:

```
[edit policy-options]
root@northstar_junosvm # show
policy-statement TE {
    from family traffic-engineering;
    then accept;
}
[edit protocols mpls traffic-engineering]
root@northstar_junosvm # show
database {
    import {
        policy TE;
    }
}
```

You can verify that both the TED and the lsdist.0 databases are being populated in the JunosVM by using the relevant show commands (the outputs were truncated here as these can be quite large, depending on the size of the network). Note how the TED is identical to the one in the devices, which you already saw earlier.

[illegible]

Since you don't expect to forward any traffic over this tunnel, you can use control plane-only traffic in both the JunosVM and the network device. Also, note that you don't need to enable family MPLS or add the GRE interface to the protocols MPLS stanza, since you shouldn't expect any MPLS traffic to be sent over this interface:

```
[edit interfaces gre]
root@northstar_junosvm # show
unit 0 {
    tunnel {
        source 192.168.10.101;
        destination 11.0.0.106;
    }
    family inet {
        address 10.100.0.2/30;
    }
}

[edit routing-options]
root@northstar_junosvm # show
static {
    route 0.0.0.0/0 next-hop 192.168.10.1;
    /* This host route may seem redundant; after all it points to the default gateway. It is
    needed so that when the tunnel comes up and you start learning routes over OSPF, you don't try
    to send traffic to the tunnel endpoint over the tunnel itself */
    route 11.0.0.106/32 next-hop 192.168.10.1;
}
autonomous-system 100;

[edit protocols ospf]
root@northstar_junosvm # show
traffic-engineering;
area 0.0.0.0 {
    interface lo0.0 {
        passive;
    }
    interface gre.0 {
        interface-type p2p;
        /* Just out of precaution this is configured a high metric to prevent the network from using
        this tunnel to forward any data traffic */with
        metric 65535;
    }
}

[edit protocols mpls traffic-engineering]
root@northstar_junosvm # show
database {
    import {
        policy TE;
    }
}
```

```
[edit policy-options]
root@northstar_junosvm # show
policy-statement TE {
    from family traffic-engineering;
    then accept;
}
```

WARNING It is a good idea to add a static host route to the tunnel endpoint (11.0.0.106), since it's possible to learn how to reach the tunnel endpoint through the tunnel itself (which would cause the tunnel to flap constantly).

The configuration on the router terminating the tunnel is almost identical (but here you don't need to leak the contents of the TED into the `lsdist.0` table). The additional tunnel and routing configuration on P6 ends up like this:

```
[edit interfaces gre]
root@P6# show
unit 0 {
    tunnel {
        source 11.0.0.106;
        destination 192.168.10.101;
    }
    family inet {
        address 10.100.0.1/30;
    }
}
[edit protocols ospf]
root@P6# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.0 {
    interface all {
        interface-type p2p;
    }
    interface gre.0 {
        interface-type p2p;
        metric 65535;
    }
}
```

You're almost done, but for this scenario there's an additional step. Since OSPF was disabled in the interface connecting P5 to the controller, the rest of the network doesn't know how to reach the 192.168.10.0/24 subnet, and P6 won't be able to reach the NorthStar Controller to establish the GRE tunnel.

This is a simple problem and you can inject a route to the 192.168.10.0/24 subnet into OSPF in a number of ways. A simple way is to enable OSPF in the `ge-0/0/8.0` interface on P5 but as a passive

interface. In this way, the 192.168.10.0/24 subnet is known to all the routers in the network but an adjacency between P5 and the controller won't be formed:

```
[edit protocols ospf]
root@P5# show
traffic-engineering;
reference-bandwidth 100g;
area 0.0.0.0 {
    interface all {
        interface-type p2p;
    }
    interface fxp0.0 {
        disable;
    }
    interface ge-0/0/8.0 {
        passive;
    }
}
```

Just as before, once the OSPF adjacency over the GRE tunnel is up, the contents of the TED should be injected into the lsdist.0 table on the JunosVM:

```
root@northstar_junosvm # run show route table lsdist.0

lsdist.0: 26 destinations, 26 routes (26 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

NODE { AS:11 Area:0.0.0.0 IPv4:11.0.0.101 OSPF:0 }/1152
    *[OSPF/10] 00:05:54
    Fictitious
NODE { AS:11 Area:0.0.0.0 IPv4:11.0.0.102 OSPF:0 }/1152
    *[OSPF/10] 00:05:54
    Fictitious
NODE { AS:11 Area:0.0.0.0 IPv4:11.0.0.103 OSPF:0 }/1152
    *[OSPF/10] 00:05:54
    Fictitious
NODE { AS:11 Area:0.0.0.0 IPv4:11.0.0.104 OSPF:0 }/1152
    *[OSPF/10] 00:05:54
    Fictitious
NODE { AS:11 Area:0.0.0.0 IPv4:11.0.0.105 OSPF:0 }/1152
    *[OSPF/10] 00:05:54
```

Chapter 3

MPLS LSP Management

Now that you have a working topology model, you can use this information to visualize the different MPLS LSPs configured in the network (and how they are routed). But the traffic engineering database does not include any information on the LSPs provisioned in the network – rather it contains information regarding:

- The different nodes in the network (and their hostnames, if available)
- Links between the different nodes and their characteristics, including:
 - Administration color
 - Traffic engineering metric
 - IGP metric
- Static, reservable, and available bandwidth (including per priority available bandwidth, based on the bandwidth model configured for difserv-based traffic engineering)
- Which SRLGs the link belongs to

Still missing in the *Day One* lab is a way to both discover the different LSPs provisioned in the network, and to allow the North-Star Controller to influence the paths that the different LSPs take. It should also be possible to configure new LSPs directly from the controller, which could be pushed into the network whenever

desired. While it is possible to use NETCONF or the CLI connections to both discover and configure LSPs, the use of such mechanisms is not without its own set of problems.

- Different vendors use different syntaxes both in their CLI and the schemas used for NETCONF. Therefore, if any such mechanism was used, it would have to be tailored to the details of each vendor's implementation.
- Neither the CLI nor the NETCONF-based connections are meant to be used to push rapidly changing configurations. Both mechanisms tend to perform multiple syntactic and semantic checks on the configuration data pushed, and in most cases they also perform change control operations. While all these are desirable properties on interfaces that are exposed to users, they tend to slow down the operations (to the point where pushing large configuration changes in a device can take several minutes to commit).
- While both NETCONF and CLI commands provide a way to monitor the status of different elements (such as the LSPs), the commands to use, as well as the format of the output, are proprietary and vendor specific.
- The NETCONF protocol does include an event notification mechanism that provides an asynchronous message delivery service that can be used to notify the controller of changes in the status of an LSP (or update the controller when the state of the network changes). Here, yet again, the schema of the different notification messages is vendor specific. Perhaps more importantly, a mechanism to synchronize the state between the different network devices and the controller is desirable so stateful controllers (such as NorthStar) that keep track of set of active LSPs, paths, and reserved resources can have an up-to-date representation of the status of the network. No such mechanism currently exists that works over NETCONF and could be used to achieve this goal.

For these reasons (and others) the Path Computation Element Communication protocol (PCEP) was designed and specified in [RFC5440](#). It defines the set of procedures required to communicate a Path Computation Element (PCE), the NorthStar Controller, in our case, and one or more Path Computation Clients or PCCs (the routers in the network that initiate LSPs). PCEP covers all the requirements previously highlighted, while designed to be easily extended to accommodate for future requirements and it is the topic of our next section.

The Path Computation Element Protocol (PCEP)

The objective of PCEP is to provide a means for communication between a PCE (the NorthStar controller in our case) and one or more PCCs.

It is instructive to consider what happens, conceptually, when a network element such as a Juniper router is configured to establish an MPLS LSP with some path constraints. When a device needs to signal an LSP, it will try to compute the path to the destination depending on the configuration:

- The LSP is configured with the no-cspf knob. In this case, the Explicit Route Object, or ERO, which details the path the LSP must take, is not computed by the head-end and the LSP will be loosely routed. Depending on the configuration the LSP might have no ERO, might contain an ERO with at least one loose hop, or it may contain an abstract node in the ERO.
- If CSPF is enabled (the default behavior), the head-end will try to compute the full path that satisfies the constraints. To do so, it will query the traffic engineering database and use the CSPF algorithm to find the best available path. Once found, the path will be used in the ERO when the LSP is signaled.
- Even with CSPF enabled, there are cases in which it isn't possible to compute a strictly routed ERO. Instead, some intermediate nodes, like ABRs and ASBRs may have to expand the ERO and figure out the path to the destination (or other ABRs/ASBRs).

In any case, if CSPF is enabled the head end device will need to figure out the best path towards the destination or border device and will do so by walking through the TED and selecting the best available path that satisfies the constraints (using the Dijkstra's algorithm).

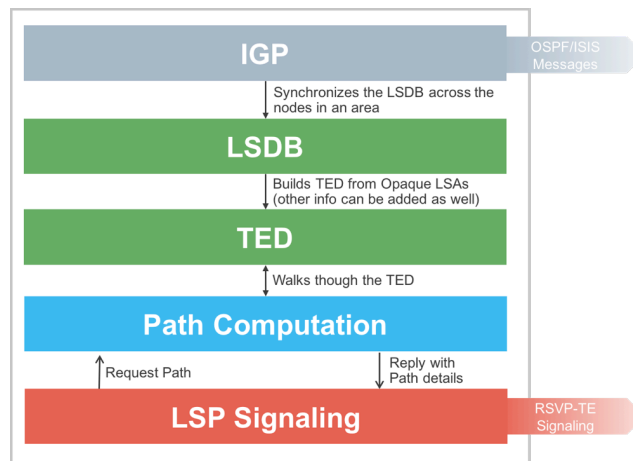


Figure 3.1 Path Computation

The main intent of PCEP is to decouple the path computation element from the network device, so that an external controller (centralized or distributed) can compute the paths. There are many motivations for this, and curious readers will do to read the [RFC 4655](#) which goes over the architecture and motivations. For now it will suffice to say that some of the more important motivations are:

- Off-loading some CPU intensive computations, for example, computing the path of a set of LSPs while minimizing some function such as the max link utilization or the aggregated capacity consumed.
- Enabling the placement of LSPs, including constraints, for cases in which the head-end nodes have only partial visibility of the network. Such is the case of inter-area or inter-as LSPs.
- Optimizing placement of LSPs for cases where a full view of all the LSPs in the network is desirable, or even required (also related to the CPU offload case). Consider, for example, the classic bin packing problem: head-end routers compute the paths of the LSPs they originate without having any visibility into LSPs originated by other devices. Because of this, the order in which the LSPs are signaled may end up determining if an LSP can be placed, even if there is enough available capacity in the network. Imagine a simple scenario, even within a single head-end, where two LSPs to some destination need to be placed and for which there are two possible paths. Path A has a maximum capacity of 10 Gbps, while path B has 5 Gbps available. Let's now imagine that our LSPs require 5 and 6Gbps respectively. If we were to signal the LSP requiring 5Gbps first and that LSP would take path A, we would be left with 5Gbps remaining in path A and 5Gbps in path B. Neither path would be capable of accommodating the 6Gbps LSP that are left to place. Instead, if we had signaled the LSPs in the reverse order, we would have been able to accommodate both LSPs in the network. This simple example illustrates the problem, which gets amplified considering that many different devices will attempt to place LSPs in the network in a non-coordinated fashion.

With PCEP you are able to disassociate the path computation operation from the network device, centralizing the computation of the paths in the NorthStar Controller. Of course, the controller needs to know the contents of the TED (or, as you saw in some of the previous examples, it can build a TED spanning multiple areas or domains) and you already saw the different ways that the NorthStar controller can obtain that information.

NOTE The PCE Architecture does specify options where multiple controllers can be utilized in a distributed manner. Let's focus on the centralized computation model, using a single controller.

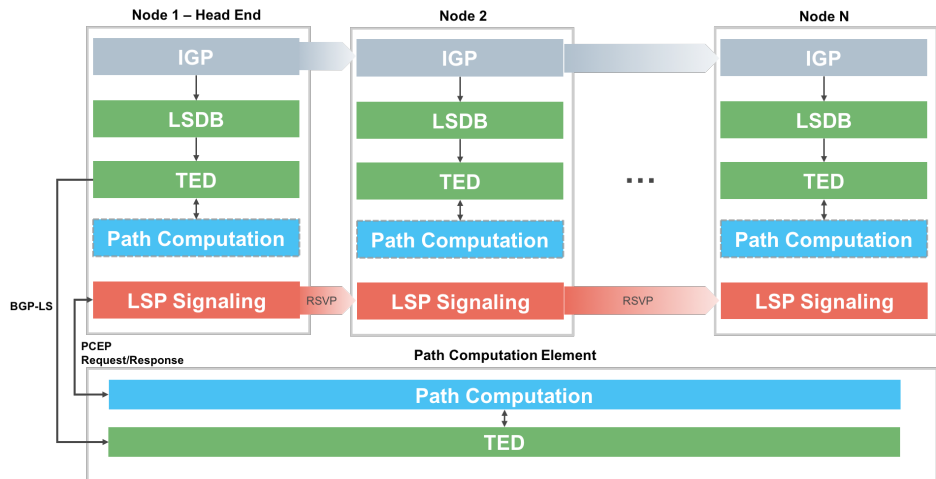


Figure 3.2 External Path Computation Element

In this light, the PCEP protocol is a request/response protocol used to communicate path computation requests from a PCC to a PCE, and send the responses back from the PCE to the PCC. In addition, the PCE architecture specifies a few different types of computation elements, which determine some of the capabilities and protocol extensions used to deliver them:

- **Stateless PCEs:** The PCE utilizes only information from the TED to calculate paths. Stateless PCEs keep no state of the LSPs in the network, and once they return a path computation they don't keep track of the different requests received or the status of the LSPs.
- **Stateful PCEs:** The PCE keeps track of the set of paths and resources utilized in the network. In this case the PCE not only uses information from the TED to place demands, but also obtains information regarding the paths, status, and resources of the existing LSPs in the network.
- **Passive PCEs:** By default, PCEs do not actively update the state of an LSP.
- **Active PCEs:** An active PCE can update the parameters of the LSPs over which it has control. Active LSPs also must be stateful, since they need to at least keep track of the LSPs that they have control over.

NorthStar is an active stateful controller, which means that it keeps track of all the LSPs initiated by all the clients it connects to, but also it can modify the placement and parameters of the LSPs it has control over. Because the state information of the various LSPs configured in a network is not synchronized across all nodes, the PCE needs to connect to all PEs (and, in general, any node initiating an LSP) to get LSP information, compute the paths of any LSPs delegated to the controller, or to provision new LSPs. To do so, you need to configure PCEP in the PE devices in your network.

Configuring PCEP

Before getting into the configuration details, you need to make sure the devices support the PCEP protocol. Support for the PCEP protocol in the base Junos image is included from Junos version 16.1 onwards. On earlier images the JSDN package is required, which includes support for PCEP and OpenFlow protocols.

For devices that do not include native support for PCEP (at this time, the Junos version 14.2R4 is qualified to work with NorthStar), install the JSDN package which can be downloaded from the Junos software repository. Make sure the package version matches the version of Junos running in your devices. The process to install a Junos package is very similar to the one used to upgrade and install a new version of Junos.

First, copy the JSDN package to the devices (choose your favorite method to copy the file such as ssh or ftp), and run the “request system software add” command to install the package.

```
root@PE1# run request system software add jsdn-i386-14.2X1.1.tgz
NOTICE: Validating configuration against jsdn-i386-14.2X1.1.tgz.
NOTICE: Use the 'no-validate' option to skip this if desired.
Checking compatibility with configuration
Initializing...
Using jbase-14.2X1.1
/* ... output truncated for brevity...*/
CLI release 14.2X1.1 built by builder on 2015-04-11 01:55:41 UTC
Restart cli using the new version ? [yes,no] (yes) yes
Restarting cli ...
```

After the CLI client is restarted, the [configuration pcep] hierarchy will be available and you'll be able to enable PCEP between the device and controller as shown in Figure 3.3.

Junos implements PCC-initiated PCEP connections, so all the configuration details and negotiation options are specified in the routers. With the default configuration, the controller will accept all connections to port 4189 (the default PCEP port) on all its interfaces, which means you don't have to make any configuration changes in the controller.

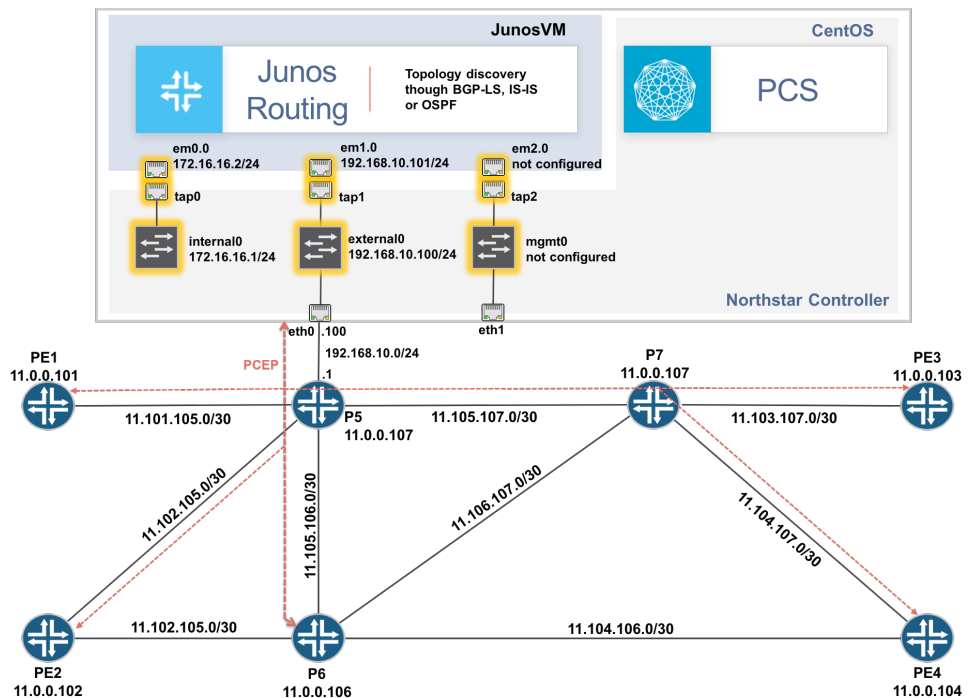


Figure 3.3 PCEP Connection to the Controller

With the JSDN package installed, just define a new PCE server and make the PCC process responsible for the management of externally-controlled LSPs. Start with PE1; then this process has to be performed in all PEs in the network:

```
[edit protocols pcep]
root@PE1# show
pce jnc {
    /* Make sure this is the loopback address of the device you are configuring */
    local-address 11.0.0.101;
    destination-ipv4-address 192.168.10.100;
    destination-port 4189;
    pce-type active stateful;

    /* This command allows the controller to be used to provision new LSPs. If this is not set,
    the controller will be able to monitor and compute the PATH for delegated LSPs, but it won't
    be able to provision new LSPs on the device */
    lsp-provisioning;
}
[edit protocols mpls]
root@PE1# show
/* This command is required so that the pccd daemon, in charge of the PCEP communication with
the controller, can be used to control LSPs as well as receive information regarding the
status of the LSPs in the device */
lsp-external-controller pccd;
```

Verify the status of the connection with the NorthStar Controller by looking at the output of the `show path-computation-client status` command:

```
root@PE1# run show path-computation-client status
```

Session	Type	Provisioning	Status
jnc	Stateful Active	On	Up

LSP Summary

```

Total number of LSPs      : 10
Static LSPs               : 10
Externally controlled LSPs : 0
Externally provisioned LSPs : 0/16000 (current/limit)
Orphaned LSPs             : 0

```

jnc (main)

```

Delegated                  : 0
Externally provisioned    : 0

```

At this point, it may be worth explaining the nomenclature used for the different types of LSPs:

- **Static LSPs:** These are plain vanilla LSPs provisioned directly in the router through the conventional mechanisms (the CLI or NETCONF). These LSPs cannot be modified by the controller but, nevertheless, their status and characteristics are reported to the controller. This way, the NorthStar Controller can not only display all the LSPs in the network, but also take into account the already provisioned LSPs (and the resources they utilize) when computing paths for LSPs that it can control.
- **Externally controlled (or delegated) LSPs:** These are LSPs that, while they have been configured in the device through the CLI or NETCONF, have had their control delegated to the NorthStar Controller. This means that the controller is used to compute the different paths the LSP will use, signal new bandwidth, priorities, etc.
- **Externally provisioned LSPs:** These LSPs have been created by the NorthStar Controller and pushed into the device through PCEP. The configuration database has no record of such LSPs, yet the LSPs will be shown when executing the `show mpls lsp` command (new modifiers have been added to display only externally provisioned LSPs, or even externally controlled LSPs).

The terminology is a bit different in the NorthStar Controller, since LSPs are named from the controller's perspective. All the names are pretty straightforward, as listed in Table 3.1.

Table 3.1 NorthStar Terminology for LSP Control Types

Description	LSP type on the device	LSP control type on the NorthStar Controller
Vanilla LSPs provisioned in the devices	Locally Provisioned	PCC
LSPs provisioned in the device and controlled by the controller	Externally controlled	Delegated
LSPs provisioned from the controller	Externally provisioned	PCE Initiated

There are some significant differences between delegated LSPs and externally provisioned LSPs, apart from the obvious fact that the provisioning of the former is done in the device while the provisioning of the latter is done from the controller.

Perhaps the more salient difference between delegated and externally provisioned LSPs is what happens after a failure in the controller. Delegated LSPs are maintained and provisioned in the devices. If a device fails to connect to a controller (due to connectivity issues or a controller failure) control of delegated LSPs is returned to the originating device, which will then compute the path just as done normally with static LSPs. The `delegation-cleanup-timeout` knob, under the PCE settings, controls how long the device waits after the PCEP session is terminated to regain control of the LSPs.

In contrast, externally provisioned LSPs are cleared if the PCEP connection to the NorthStar controller is lost. The `lsp-cleanup-timer` knob (available under the `pce <controller name>` settings) controls how long the device will wait (in seconds) to clean up externally provisioned LSPs after a failure. The default value is 0, which means that the LSPs will be cleared immediately after a controller failure.

WARNING

Externally provisioned LSPs are not provisioned in the configuration database in the Junos OS. Configuration changes pushed through PCEP are applied at a much higher rate than configuration changes done through the configuration database. However, the configuration changes are not persistent after a device is rebooted. If a controller and all its backup controllers fail and, at the same time, network devices fail, all externally provisioned LSPs in the failed devices will be cleared. Only after the controller reconnects to the failed devices will those LSPs be re-provisioned.

As the nodes start exchanging information with the controller using PCEP, information regarding the LSPs in the network will start being synchronized as shown in Figure 3.4.

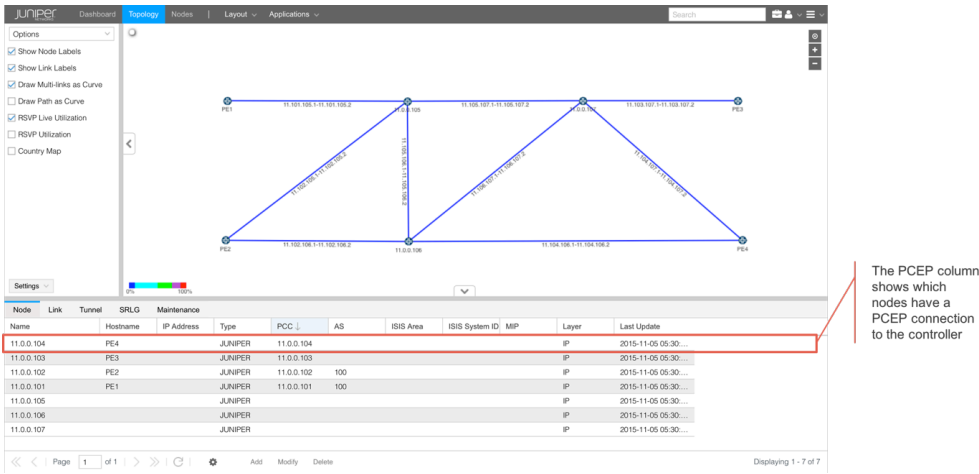


Figure 3.4 Controller Nodes Table

Which LSPs will each node report? The ones they originate, of course! Although transit and egress nodes have some visibility into the LSPs they carry, most of the state of an LSP is kept in the head-end nodes, which signaled the LSPs to begin with. If, for example, an LSP has several paths, transit nodes will only have knowledge of the paths they carry (same applies to sub-LSPs on P2MP LSPs). In the end, client nodes (the routers talking to the NorthStar Controller) only publish information of the LSPs they originate.

Visualizing LSPs

PCEP allows stateful controllers like the NorthStar Controller to discover all LSPs configured in the nodes it connects to (regardless of how those LSPs have been provisioned). The Tunnels tab in the Topology view of the NorthStar GUI lists all LSPs discovered through PCEP in the network. If you right click when selecting nodes, links allow you to filter the list of tunnels, displaying only tunnels initiated or terminated in the selected nodes, or that happen to transit a given group of links.

When an active LSP is selected, the path it takes in the network is highlighted (live path) as shown in Figure 3.5.

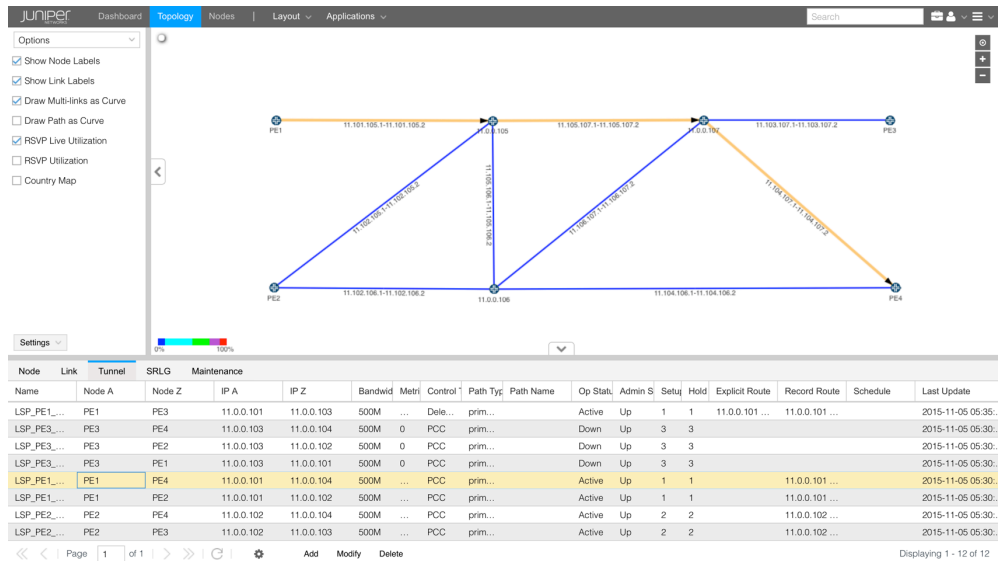


Figure 3.5 LSP Path View

LSP Management Basics

By default, LSPs configured in the devices are static – they can be viewed but can’t be modified by the NorthStar Controller. All the path computations are done on static LSPs by the head-end routers where the LSPs are configured. When LSPs are delegated to a controller, the controller can not only modify the LSP’s configuration settings, but it can also be used for path computations.

To delegate an LSP to a controller, you need to add the `lsp-external-controller pccd` knob in the LSP configuration. With it, the LSP will be configured in the device but the path computation will be delegated to the controller. After a controller failure, Constrained Shortest Path First (CSPF) will be run by the device as it would have been done if the LSP were static.

Furthermore, since the controller is in charge of calculating the paths (as well as some of the LSP attributes) that delegated LSPs will take, and the controller has a full view of the network, inter-area LSPs can be managed without much fanfare. You may have noticed that after you changed your network configuration to include a new OSPF area, all the LSPs that crossed area boundaries went down (any LSP originating or terminating on PE3) – the routers could not find the path to the destination because they didn’t have the topology information of the nodes outside their own area. This should be clear by looking at the tunnels table shown in Figure 3.6.

Node	Link	Tunnel	SRLG	Maintenance														
Name	Node A	Node Z	Bandwidth	Metric	Control Type	Path Type	Path Name	Op Status	Admin S	Setup	Hold	Explicit Route	Record Route	Schedule	Last Update			
LSP_PE2_PE3	PE2	PE3	500M	300	PCC	primary		Active	Up	2	2		11.0.0.102 ...		2015-11-05 05:30...			
LSP_PE2_PE1	PE2	PE1	500M	200	PCC	primary		Active	Up	2	2		11.0.0.102 ...		2015-11-05 05:30...			
LSP_PE4_PE2	PE4	PE2	500M	200	PCC	primary		Active	Up	4	4		11.0.0.104 ...		2015-11-05 05:30...			
LSP_PE4_PE1	PE4	PE1	500M	300	PCC	primary		Active	Up	4	4		11.0.0.104 ...		2015-11-05 05:30...			
LSP_PE3_PE4	PE3	PE4	500M	0	PCC	primary		Down	Up	3	3				2015-11-05 05:30...			
LSP_PE3_PE2	PE3	PE2	500M	0	PCC	primary		Down	Up	3	3				2015-11-05 05:30...			
LSP_PE3_PE1	PE3	PE1	500M	0	PCC	primary		Down	Up	3	3				2015-11-05 05:30...			
LSP_PE4_PE3	PE4	PE3	500M	0	PCC	primary		Down	Up	4	4				2015-11-05 05:30...			

<< < | Page 1 of 1 > >> | ⌂ ⚙ Add Modify Delete

Displaying 1 - 12 of 12

Figure 3.6 Tunnels Table

As you make those LSPs delegated, the NorthStar Controller will find a suitable path and the LSPs will come up. Let's do that now for the LSP between PE1 and PE3:

```
[edit protocols mpls]
root@PE1# show
label-switched-path LSP_PE1_PE3 {
    to 11.0.0.103;
    bandwidth 500m;
    priority 1 1;
    adaptive;
/* This is the important part, the configuration of this LSP is just like any other but the
LSP's control will be delegated to an external pcep controller if available */
    lsp-external-controller pccd;
}
```

In the GUI, delegated LSPs are indicated with the *Delegated* LSP control type in the tunnels table as shown in Figure 3.7. You can also see that the path goes across two areas.

LSPs that are either delegated to the controller or controller-provisioned can be modified by clicking the Modify Selected button in the Tunnels tab, after selecting one or more tunnels from the table. Only the following attributes can be modified from the controller for delegated LSPs:

- Bandwidth
- Setup and hold priorities
- ERO that is, the LSP's path

NOTE At the time of this writing, some of these parameters are only configurable through the Java client.

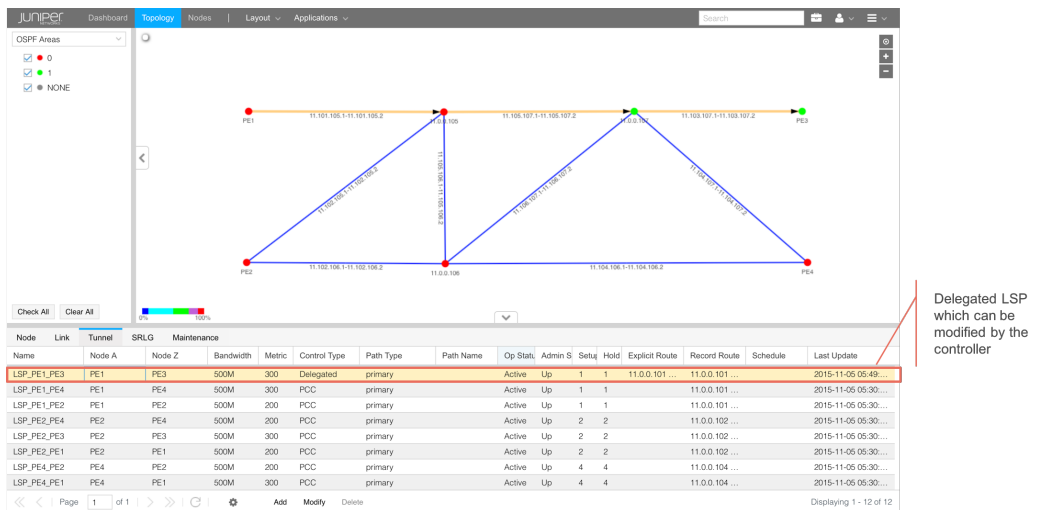


Figure 3.7 Delegated LSPs

While an LSP is managed by a controller, CLI configuration changes made to the LSP will be stored in the configuration file, but won't take effect until control of the LSP is returned to the device.

NOTE When using the Java client, changes made to the network are not pushed automatically, but rather stored locally in the controller. LSPs with changes are marked in the UI with PENDING status. Simply select the LSPs you want to push changes to and click the Provision Selected button, to push the pending changes into the network.

Last, but not least, you can create new LSPs directly from the NorthStar Controller's UI. From the Applications drop-down menu, select the Provision LSP option. A pop-up menu will show the tunnel parameters that can be configured through the NorthStar Controller. Here you can:

- Type a tunnel ID (any string that will identify the tunnel)
- Specify the tunnel properties like the bandwidth, setup, and hold priorities, etc.
- Select the endpoints from the Node A and Node Z drop downs (you can also right-click on a node in the UI and select the "Add LSP from this node" option).
- Click the Submit button.

The screenshot shows a web-based form titled "Add Tunnel". It has two tabs: "Properties" and "Scheduling". The "Properties" tab is active. The form contains the following fields and controls:

- Name:** A text input field containing "MyLSP".
- Node A:** A dropdown menu showing "PE1".
- Node Z:** A dropdown menu showing "PE4".
- Bandwidth:** A text input field containing "4G".
- Coloring:** An empty text input field.
- Setup:** A numeric input field containing "7".
- Hold:** A numeric input field containing "7".
- Comment:** A large text area.
- Buttons:** At the bottom, there are three buttons: "Preview Path" (disabled), "Cancel" (disabled), and "Submit" (active).

Figure 3.8 Adding LSPs from the Controller

NOTE It's instructive to play around with the different tunnel properties and see how they affect the tunnel placement. Use the Display Path button to view how the tunnel will be routed, based on the constraints and properties specified, even before adding the tunnel.

When using the Java client don't forget to provision the tunnel if you want the changes to be pushed to the network. If the *RSVP live utilization view* is selected, as you add new LSPs, the link colors will change according to their utilization, making it really easy to identify hot-spots in the network.

LSP Templates

When configuring LSPs, there are some configuration options that are either not yet part of the PCEP specification or that could be vendor-specific. Naturally, new options and vendor-specific extensions to PCEP will cover most of these configuration settings, but it is likely that there will always be some advanced or not-so-common settings that won't be configurable through PCEP.

The Junos OS solves this problem though the use of LSP templates. The idea is quite simple – you can create an LSP template that will be applied to the LSPs provisioned from the controller. For example, let's say you want to add an LSP with link protection configured. This configuration option isn't available yet on the controller (at the time of this writing, the IETF is working on PCEP extensions to deal with local protection mechanisms), so you can create an LSP template with link-protection enabled and specify that this is the default template to use for externally provisioned LSPs. Here's how:

```
[edit protocols mpls]
root@PE1# show
lsp-external-controller pccd {
    label-switched-path-template {
        LP-template;
    }
}
label-switched-path LP-template {
    template;
    link-protection;
}
```

After adding a few LSPs from the controller you can see that all externally-provisioned LSPs have the template applied:

```
root@PE1# run show path-computation-client lsp
```

Name	Status	PLSP-Id	LSP-Type	Controller	Template
LSP_PE1_PE4	Primary(Act)	2	local	-	
LSP_PE1_PE3	Primary(Act)	3	ext-controlled	jnc	
LSP_PE1_PE2	Primary(Act)	4	local	-	
LSPtest	Primary(Act)	5	ext-provided	jnc	LP-template

But what if you want to configure different parameters for different LSPs? In other words, you need a way to select which template to apply to a group of LSPs. To do this, you can add *mapping rules* so that externally provisioned LSPs with names matching a configurable pattern (specified with a regular expression) will use a given template.

For example, let's say you want to add an LSP with node-link protection configured. You can enforce a naming convention so that any LSP containing *NLP_* in their name will be configured with node link protection. And here's the configuration to do this:

```
[edit protocols mpls]
root@PE1# show
lsp-external-controller pccd {
    pce-controlled-lsp NLP_* {
        label-switched-path-template {
            NLP-template;
        }
    }
}
label-switched-path NLP-template {
    template;
    node-link-protection;
}
```

To test the configuration, add a new LSP with its name starting with *NLP_*, and if you are following this book in your lab, add an LSP from PE1 to PE2 named *NLP_PE1_PE2*. From the `show path-computation-client lsp` output, you'll see which template was applied to each LSP.

Take a moment and play around here. It's kind of fun and instructive at the same time.

Going Forward

There are many valuable use cases that can be accommodated by using the NorthStar Controller to either optimize existing paths or provision new paths. While it's outside the scope of this *Day One* book to go into the details of how each of these use cases can be addressed, let's at least review a list with the most common and valuable use cases so when you have time to experiment with the NorthStar Controller you know right where to go.

Optimize existing paths or provision new paths to:

- Visualize the network, find links not configured for MPLS, or where routing adjacencies aren't being formed.
- Provision LSPs including single, one-to-many, many-to-many, and full mesh tunnels.
- Place tunnels based on user configurable metrics such as link delays, geographical distance, IGP metrics, hop count, etc.
- Monitor the status of the network in real-time, including the status of nodes, links, and LSPs.
- View and correlate node, link, and tunnel events.
- View the history of the LSPs over time.
- Perform path analysis, where the controller can discover sub-optimal paths and make the required changes.
- Periodically optimize the network.
- Provision diverse paths that avoid common link or sites in order to reduce the chances of simultaneous failures.
- Placement and monitoring of auto-bandwidth and TE++ LSPs.
- Put nodes, links, and sites in maintenance mode, where LSPs are routed around elements entering a maintenance window.

Summary

By now you should be able to install a controller, connect it to the network, configure the devices to export the topology, establish a PCEP connection to the controller, and finally, use the controller to visualize and provision LSPs.

But what to do when things go wrong? There are many moving parts in the solution, so it is instructive to go through the whole chain from the devices, all the way to the controller, to check the status of the different protocols and connections. And that, indeed, is the topic of the next chapter.

Chapter 4

Troubleshooting

When troubleshooting issues, it's helpful to follow a systematic approach in order to narrow down the problem. This chapter provides guidance on how to isolate problems with the NorthStar Controller and your network. It should be both instructive and useful when troubleshooting any potential issues. Don't put it off until after your incidents occur.

Let's start from the devices and move all the way to the NorthStar Controller, looking at the status of the various protocols along the way.

Topology Acquisition

First, let's look into the device (or devices) exporting the topology.

1. Are the contents of the traffic engineering database (TED) being correctly populated in the devices exporting the topology to the controller?

```
root@P5> show ted database
TED database: 0 ISIS nodes 6 INET nodes
ID                                     Type Age(s) LnkIn LnkOut Protocol
11.0.0.101                             Rtr   268      1      1 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.101.105.1, Remote: 11.101.105.2
    Local interface index: 330, Remote interface index: 0
ID                                     Type Age(s) LnkIn LnkOut Protocol
11.0.0.102                             Rtr   256      2      2 OSPF(0.0.0.0)
  To: 11.0.0.106, Local: 11.102.106.1, Remote: 11.102.106.2
    Local interface index: 331, Remote interface index: 0
  To: 11.0.0.105, Local: 11.102.105.1, Remote: 11.102.105.2
    Local interface index: 330, Remote interface index: 0
ID                                     Type Age(s) LnkIn LnkOut Protocol
11.0.0.104                             Rtr   249      2      2 OSPF(0.0.0.0)
  To: 11.0.0.106, Local: 11.104.106.1, Remote: 11.104.106.2
    Local interface index: 330, Remote interface index: 0
  To: 11.0.0.107, Local: 11.104.107.1, Remote: 11.104.107.2
    Local interface index: 331, Remote interface index: 0
...
```

If the TED is empty, check the IGP's link-state database (LSDB) to make sure that you can see all the relevant links state advertisements:

```
root@P5# run show ospf database
  OSPF database, Area 0.0.0.0
  Type      ID          Adv Rtr      Seq          Age  Opt  Cksum  Len
Router     11.0.0.101      11.0.0.101  0x80000002   965  0x22 0x468d  60
Router     11.0.0.102      11.0.0.102  0x80000003   935  0x22 0x7f3c  84
Router     11.0.0.104      11.0.0.104  0x80000003   929  0x22 0x5552  84
Router     *11.0.0.105      11.0.0.105  0x80000005   940  0x22 0x435c 144
Router     11.0.0.106      11.0.0.106  0x80000005   928  0x22 0x6312 132
Router     11.0.0.107      11.0.0.107  0x80000004   944  0x22 0xcdaf 108
Summary    11.0.0.103      11.0.0.107  0x80000001   956  0x22 0xf3fb  28
Summary    11.103.107.0    11.0.0.107  0x80000003   227  0x22 0x7114  28
OpaqueArea 1.0.0.1         11.0.0.101  0x80000001   977  0x22 0xaba2  28
...
```

Or, if you are using IS-IS as your IGP:

```
root@P5# run show isis database
IS-IS level 1 link-state database:
  0 LSPs

IS-IS level 2 link-state database:
LSP ID      Sequence Checksum Lifetime Attributes
PE1.00-00   0x2      0x842    886   L1 L2
PE1.02-00   0x2      0x6d22   886   L1 L2
PE2.00-00   0x5      0x4ad9   1186  L1 L2
PE2.02-00   0x2      0x6e1f   887   L1 L2
PE2.03-00   0x1      0x7d0f   1186  L1 L2
P5.00-00    0xc      0xef5a   1189  L1 L2
P5-p106.00-00 0x5      0x6f79   1184  L1 L2
P5-p107.00-00 0x5      0x5c2e   1187  L1 L2
P5-p107.02-00 0x1      0x750f   1187  L1 L2
  9 LSPs
```

If the LSDB is being correctly populated but the TED is empty, you'll have to make sure that traffic engineering extensions have been enabled in the IGP. IS-IS enables traffic engineering extensions by default, but it could have been disabled with the `set protocols isis traffic-engineering disable` command. With OSPF one has to enable traffic engineering extensions explicitly with the `set protocols ospf traffic-engineering` command.

If the LSDB is empty, or there are missing nodes, it is likely due to a configuration problem with the IGP.

2. If the TED is being correctly populated, the next step is to check the JunosVM to make sure that the contents of the TED are being imported into the `lsdist.0` database. Here, you have two possibilities:

- When using BGP-LS to export the contents of the TED, you need to check that BGP is up and that you are learning the traffic engineering information from the peer. Check that the BGP peering session between the JunosVM and the devices exporting the topology is up (do this from the JunosVM):

```
# run show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Unconfigured peers: 2
Table Tot Paths Act Paths Suppressed History Damp State Pending
lsdist.0
Peer 38 AS 38 InPkt 0 OutPkt 0 OutQ 0 Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
11.0.0.106 100 194 110 0 0 47:48 Establ
lsdist.0: 17/17/17/0
11.0.0.107 100 16 2 0 0 2 Establ
lsdist.0: 21/21/21/0
```

■ If BGP is not up, make sure that the BGP endpoints have IP connectivity and that BGP is correctly configured in the device connecting to the JunosVM (remember you should not need to manually change the BGP configuration in the JunosVM). Also, make sure that the AS numbers are correctly configured.

Once you are sure that the BGP sessions between the JunosVM and the devices exporting the topology are up, make sure you are receiving the topology information from the peers:

```
# run show route receive-protocol bgp 11.0.0.106

inet.0: 46 destinations, 46 routes (46 active, 0 holddown, 0 hidden)

mpls.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

lsdist.0: 38 destinations, 38 routes (38 active, 0 holddown, 0 hidden)
Prefix Nexthop MED Lc1pref AS path
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.101 OSPF:0 }/1152
* 11.0.0.106 100 I
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.102 OSPF:0 }/1152
* 11.0.0.106 100 I
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.104 OSPF:0 }/1152
* 11.0.0.106 100 I
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.106 OSPF:0 }/1152
* 11.0.0.106 100 I
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.107 OSPF:0 }/1152
* 11.0.0.106 100 I
LINK { Local { AS:100 Area:0.0.0.0 IPv4:11.0.0.101 }.
{ IPv4:11.101.105.1 } Remote { AS:100 Area:0.0.0.0 IPv4:11.0.0.105 }.
{ IPv4:11.101.105.2 } OSPF:0 }/1152
* 11.0.0.106 100 I
LINK { Local { AS:100 Area:0.0.0.0 IPv4:11.0.0.102 }.
{ IPv4:11.102.105.1 } Remote { AS:100 Area:0.0.0.0 IPv4:11.0.0.105 }.
{ IPv4:11.102.105.2 } OSPF:0 }/1152
...
```

You can also directly check the contents of the lsdist.0 table, which contain all the routes imported – there shouldn't be any import policy filtering the BGP routes received at the JunosVM, but it is still a good idea to make sure that the routes are making it into the lsdist.0 table:

```
# run show route table lsdist.0

lsdist.0: 38 destinations, 38 routes (38 active, 0 holddown, 0 hidden)
```

+ = Active Route, - = Last Active, * = Both

```

NODE { Area:0.0.0.0 IPv4:11.0.0.101 OSPF:0 }/1152
    *[BGP/170] 00:08:18, localpref 100, from 11.0.0.107
        AS path: I, validation-state: unverified
        > to 192.168.10.1 via em1.0
NODE { Area:0.0.0.0 IPv4:11.0.0.102 OSPF:0 }/1152
    *[BGP/170] 00:08:18, localpref 100, from 11.0.0.107
        AS path: I, validation-state: unverified
        > to 192.168.10.1 via em1.0
NODE { Area:0.0.0.0 IPv4:11.0.0.104 OSPF:0 }/1152
    *[BGP/170] 00:08:18, localpref 100, from 11.0.0.107
        AS path: I, validation-state: unverified
        > to 192.168.10.1 via em1.0
NODE { Area:0.0.0.0 IPv4:11.0.0.106 OSPF:0 }/1152
    *[BGP/170] 00:08:18, localpref 100, from 11.0.0.107
        AS path: I, validation-state: unverified
        > to 192.168.10.1 via em1.0

```

If BGP is up but you aren't getting any routes, the problem is likely the BGP configuration in the devices exporting the topology. Make sure that address family traffic-engineering is enabled and that an export policy matching on family traffic-engineering, with an accept action, is applied to the BGP session (the configuration is shown in Chapter 1 right after Figure 1.9).

If IS-IS or OSPF are used to propagate the traffic-engineering database to the controller directly, then make sure that the routing adjacency is up:

```

# run show ospf neighbor
Address      Interface      State      ID            Pri  Dead
192.168.10.1 em1.0          Full      11.0.0.105    128  34

```

Check that the TED is being directly built in the JunosVM (you already checked that the TED is being built in the device exporting the topology):

```

# run show ted database
TED database: 0 ISIS nodes 8 INET nodes
ID              Type Age(s) LnkIn LnkOut Protocol
11.0.0.101      Rtr   131     1     1 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.101.105.1, Remote: 11.101.105.2
  Local interface index: 330, Remote interface index: 0
ID              Type Age(s) LnkIn LnkOut Protocol
11.0.0.102      Rtr   131     2     2 OSPF(0.0.0.0)
  To: 11.0.0.105, Local: 11.102.105.1, Remote: 11.102.105.2
  Local interface index: 330, Remote interface index: 0
  To: 11.0.0.106, Local: 11.102.106.1, Remote: 11.102.106.2
  Local interface index: 331, Remote interface index: 0

```

Just like before, if the TED is not being populated, make sure that the traffic engineering extensions are enabled in the IGP configuration in the JunosVM.

If the TED is correct, check the contents of the `lsdist.0` table:

```
# run show route table lsdist.0
lsdist.0: 27 destinations, 27 routes (27 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.101 OSPF:0 }/1152
      *[OSPF/10] 00:00:05
      Fictitious
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.102 OSPF:0 }/1152
      *[OSPF/10] 00:00:05
      Fictitious
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.103 OSPF:0 }/1152
      *[OSPF/10] 00:00:05
      Fictitious
NODE { AS:100 Area:0.0.0.0 IPv4:11.0.0.104 OSPF:0 }/1152
      *[OSPF/10] 00:00:05
      Fictitious
```

Remember, when peering the JunosVM with an IGP you need to explicitly import the traffic engineering information propagated by the IGP into the `lsdist.0` table. If the `lsdist.0` table is empty, but the traffic engineering database has the information required, the problem is likely the lack of an import policy in the MPLS traffic engineering database (the configuration was in the section titled *Using OSPF or IS-IS*).

3. Lastly, check that the topology is being exported from the JunosVM to the topology process in the NorthStar Controller. A connection to port 450 from the topology process to the JunosVM should be present:

```
root@vrr-11# run show system connections | match 450
tcp4      0  0  172.16.16.2.450      172.16.16.1.56888    ESTABLISHED
tcp4      0  0  *.450                *.*                  LISTEN
```

If this connection is not up, you'll need to verify the status of the topology server in the controller and the topology export process in the JunosVM. The topology export process is enabled by default in the JunosVM (and you'll see a `set protocols topology-export` statement in the default configuration):

```
/* In Northstar version 2.0 and higher */
[root@northstar ~]# supervisorctl status northstar:toposerver
northstar:toposerver      RUNNING    pid 5822, uptime 4:34:53
```

PCEP

The PCEP connection between the controller and devices (PCCs) is relatively independent of the topology acquisition. Of course, in order to compute paths and perform analysis of the network, the NorthStar controller needs to understand the topology, but the status of the PCEP connection does not depend on the status of the topology export. (By the way, the status of the LSPs controlled or provisioned by the controller will depend on both topology and PCEP processes.)

1. The first thing to look at, when dealing with PCEP, is the status of the connection from all the PCCs to the PCE:

```
root@PE1> show path-computation-client status
```

```

Session      Type      Provisioning  Status
jnc          Stateful Active       On       Up

LSP Summary
  Total number of LSPs      : 4
  Static LSPs               : 2
  Externally controlled LSPs : 1
  Externally provisioned LSPs : 1/16000 (current/limit)
  Orphaned LSPs            : 0
jnc (main)
  Delegated                 : 2
  Externally provisioned    : 1

```

If the connection is not up, check that both the PCC can ping the PCE, and that the PCE server process is up in the controller:

```
[root@northstar ~]# supervisorctl status northstar:pceserver
northstar:pceserver          RUNNING   pid 5815, uptime 4:50:27
```

2. If the PCEP connection is up, we can check the status of the LSPs using the `show mpls lsp` command, which has been enhanced to include filtering options to display externally controller and externally provisioned LSP.

```

root@PE1> show mpls lsp externally-provisioned
Ingress LSP: 4 sessions
To          From          State Rt P  ActivePath  LSPname
11.0.0.104  11.0.0.101  Up    0 *          Controller-provisioned-LSP1
Total 1 displayed, Up 1, Down 0
/* Note how externally provisioned LSP are also externally controlled */
root@PE1> show mpls lsp externally-controlled
Ingress LSP: 4 sessions
To          From          State Rt P  ActivePath  LSPname
11.0.0.103  11.0.0.101  Up    0 *          LSP_PE1_PE3
11.0.0.104  11.0.0.101  Up    0 *          Controller-provisioned-LSP1
Total 2 displayed, Up 2, Down 0
Egress LSP: 2 sessions
Total 0 displayed, Up 0, Down 0
Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

```

The `show path-computation-client lsp` command displays the set of LSPs that are reported to the PCE controller and as well as their status:

```

root@PE1# run show path-computation-client lsp
Name                Status      PLSP-Id LSP-Type  Controller Template
LSP_PE1_PE3        Primary(Act) 1    ext-ctrl'd  jnc
LSP_PE1_PE4        Primary(Act) 2    local      -
LSP_PE1_PE2        Primary(Act) 3    local      -
Controller-provisioned-LSP1 Primary(Act) 4    ext-provised jnc    default_pvc

```

3. Problems with an individual externally controlled LSP can be diagnosed with the `show mpls lsp name <LSP NAME>` command, just like with any other LSP. For example, the following externally controlled LSP was configured to reserve more capacity than what's available in

the network. You can see from the `show lsp` command that the controller is not providing a route (no such route exists that satisfies the constraints):

```
root@PE1# run show mpls lsp name Problematic_LSP_PE1_PE4 extensive
Ingress LSP: 4 sessions
11.0.0.104
  From: 0.0.0.0, State: Dn, ActiveRoute: 0, LSPname: Problematic_LSP_PE1_PE4
  ActivePath: (none)
  LSPtype: Externally controlled, Penultimate hop popping
  LSP Control Status: Externally controlled
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary                               State: Dn
    Priorities: 1 1
    Bandwidth: 50Gbps
    OptimizeTimer: 900
    SmartOptimizeTimer: 180
    No computed ERO.
  12 Oct 26 16:07:15.817 EXTCTRL LSP: Sent Path computation request and LSP status
  11 Oct 26 16:07:15.817 EXTCTRL_LSP: Computation request/
lsp status contains: signalled bw 500000000000 priority setup 1 hold 1
  10 Oct 26 16:06:47.267 EXTCTRL LSP: Sent Path computation request and LSP status
  9 Oct 26 16:06:47.267 EXTCTRL_LSP: Computation request/
lsp status contains: signalled bw 500000000000 priority setup 1 hold 1
  8 Oct 26 16:06:18.567 EXTCTRL LSP: Sent Path computation request and LSP status
```

Controller Processes

Although nothing needs to be configured in the NorthStar Controller (apart from the network setup), you may occasionally have to troubleshoot issues with some of its processes. Here are a few things to watch for:

1. First make sure that all processes are up and running:

```
# supervisorctl status
infra:cassandra          RUNNING pid 1805, uptime 14:43:47
infra:ha_agent           RUNNING pid 31114, uptime 9:39:21
infra:haproxy            RUNNING pid 1800, uptime 14:43:47
infra:keepalived         RUNNING pid 31258, uptime 9:37:44
infra:nodejs             RUNNING pid 1801, uptime 14:43:47
infra:rabbitmq           RUNNING pid 1803, uptime 14:43:47
infra:zookeeper          RUNNING pid 1802, uptime 14:43:47
listener1:listener1_00   RUNNING pid 1799, uptime 14:43:47
northstar:mladapter      RUNNING pid 31239, uptime 9:37:45
northstar:npat           RUNNING pid 31250, uptime 9:37:45
northstar:npat_ro        RUNNING pid 31226, uptime 9:37:45
northstar:pcserver       RUNNING pid 31130, uptime 9:38:35
northstar:pcserver       RUNNING pid 31160, uptime 9:38:25
northstar:toposerver     RUNNING pid 31230, uptime 9:37:45
```

You can restart an individual process with the following command:

```
# supervisorctl restart northstar:pcserver
northstar:pcserver: stopped
northstar:pcserver: started
```

The logs files from the controller processes are stored under the `/opt/northstar/logs` folder. There, you'll find the logs for some of the core processes such as listed here in Table 4.1.

Table 4.1 Log Types for Core Processes

Process Name	Description	Log file
PCServer	In charge of the path computation, optimizations, communication with the UI, etc.	/opt/northstar/logs/pcs.log
Toposerver	Processes the topology information from the JunosVM, builds a common data-model, stores topology information in the database, and handles topology queries and notifications	/opt/northstar/logs/toposerver.log
PCEServer	Handles the PCEP connection with the network devices	/var/log/jnc/pcep_server.log
NodeJS	Web application framework, provides the backend for the user interface, REST API	Each application can manage its own logs. The REST server logs into /opt/northstar/logs/rest_api.log
RabbitMQ	An opensource message broker used by the different daemons to send and receive messages	opt/northstar/thirdparty/rabbitmq/ var/log/rabbitmq/ rabbit@<hostname>.log

NOTE If any of the services fail to start, check the relevant log files.

Conclusion

Managing and operating medium to large IP/MPLS networks in a timely and efficient way requires the ability to constantly supervise the status of the network, quickly react to changes, and use complex algorithms to optimize the network. The NorthStar Controller solves these and other problems by implementing real-time discovery of the network topology and LSPs. Then it uses this information to solve a number of complex operational issues such as the provisioning, placement, modification, and analysis of LSPs, as well as the monitoring and visualization of the network in an intuitive way.

Using a central controller with global visibility opens up a number of use cases that can't be created otherwise. Hopefully this *Day One* book has given you a sample taste of what lies ahead if you spend some time exploring the NorthStar GUI, because the NorthStar Controller has the base building blocks required to address an ever-expanding set of modern network demands.

Happy controlling!

Appendix

Final Configuration of the Network

The configurations for all the devices used in this book's network, after going through all the different examples, is captured here for your lab usage. This book's landing page contains the same configuration files in a convenient .rtf file for cut and pasting... see <http://www.juniper.net/dayone>.

PE1

```
set version 14.2X1.1
set system host-name PE1
set interfaces ge-0/1/1 unit 0 family inet address 11.101.105.1/30
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.101/32 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options static route 10.161.0.0/16 next-hop 10.92.63.254
set routing-options router-id 11.0.0.101
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 40g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/1.0 bandwidth 40g
set protocols mpls lsp-external-controller pccd
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE1_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE1_PE2 bandwidth 500m
```

```

set protocols mpls label-switched-path LSP_PE1_PE2 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE2 adaptive
set protocols mpls label-switched-path LSP_PE1_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE1_PE3 bandwidth 500m
set protocols mpls label-switched-path LSP_PE1_PE3 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE3 adaptive
set protocols mpls label-switched-path LSP_PE1_PE3 lsp-external-controller pccd
set protocols mpls label-switched-path LSP_PE1_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE1_PE4 bandwidth 500m
set protocols mpls label-switched-path LSP_PE1_PE4 priority 1 1
set protocols mpls label-switched-path LSP_PE1_PE4 adaptive
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/1/1.0 srlg srlg-100
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols pcep pce jnc local-address 11.0.0.101
set protocols pcep pce jnc destination-ipv4-address 192.168.10.100
set protocols pcep pce jnc destination-port 4189
set protocols pcep pce jnc pce-type active
set protocols pcep pce jnc pce-type stateful
set protocols pcep pce jnc lsp-provisioning

```

PE2

```

set version 14.2X1.1
set system host-name PE2
set interfaces ge-0/1/2 unit 0 family inet address 11.102.105.1/30
set interfaces ge-0/1/2 unit 0 family mpls
set interfaces ge-0/1/3 unit 0 family inet address 11.102.106.1/30
set interfaces ge-0/1/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.102/32
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options static route 10.161.0.0/16 next-hop 10.92.63.254
set routing-options router-id 11.0.0.102
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface ge-0/1/3.0 bandwidth 40g
set protocols rsvp interface fxp0.0 disable
set protocols mpls lsp-external-controller pccd
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE2_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE2_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE2_PE1 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE1 adaptive
set protocols mpls label-switched-path LSP_PE2_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE2_PE3 bandwidth 500m

```

```

set protocols mpls label-switched-path LSP_PE2_PE3 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE3 adaptive
set protocols mpls label-switched-path LSP_PE2_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE2_PE4 bandwidth 500m
set protocols mpls label-switched-path LSP_PE2_PE4 priority 2 2
set protocols mpls label-switched-path LSP_PE2_PE4 adaptive
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/1/2.0 srlg srlg-100
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols pcep pce jnc local-address 11.0.0.102
set protocols pcep pce jnc destination-ipv4-address 192.168.10.100
set protocols pcep pce jnc destination-port 4189
set protocols pcep pce jnc pce-type active
set protocols pcep pce jnc pce-type stateful
set protocols pcep pce jnc lsp-provisioning

```

PE3

```

set version 14.2X1.1
set system host-name PE3
set interfaces ge-0/1/8 unit 0 family inet address 11.103.107.1/30
set interfaces ge-0/1/8 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.103/32
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options static route 10.161.0.0/16 next-hop 10.92.63.254
set routing-options router-id 11.0.0.103
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/8.0 bandwidth 40g
set protocols mpls lsp-external-controller pccd
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE3_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE3_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE1 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE1 adaptive
set protocols mpls label-switched-path LSP_PE3_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE3_PE2 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE2 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE2 adaptive
set protocols mpls label-switched-path LSP_PE3_PE4 to 11.0.0.104
set protocols mpls label-switched-path LSP_PE3_PE4 bandwidth 500m
set protocols mpls label-switched-path LSP_PE3_PE4 priority 3 3
set protocols mpls label-switched-path LSP_PE3_PE4 adaptive
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable

```

```

set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.1 interface all interface-type p2p
set protocols ospf area 0.0.0.1 interface fxp0.0 disable
set protocols pcep pce jnc local-address 11.0.0.103
set protocols pcep pce jnc destination-ipv4-address 192.168.10.100
set protocols pcep pce jnc destination-port 4189
set protocols pcep pce jnc pce-type active
set protocols pcep pce jnc pce-type stateful
set protocols pcep pce jnc lsp-provisioning

```

PE4

```

set version 14.2X1.1
set system host-name PE4
set interfaces ge-0/1/7 unit 0 family inet address 11.104.106.1/30
set interfaces ge-0/1/9 unit 0 family inet address 11.104.107.1/30
set interfaces lo0 unit 0 family inet address 11.0.0.104/32 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options static route 10.161.0.0/16 next-hop 10.92.63.254
set routing-options router-id 11.0.0.104
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/7.0 bandwidth 40g
set protocols mpls lsp-external-controller pccd
set protocols mpls optimize-timer 900
set protocols mpls label-switched-path LSP_PE4_PE1 to 11.0.0.101
set protocols mpls label-switched-path LSP_PE4_PE1 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE1 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE1 adaptive
set protocols mpls label-switched-path LSP_PE4_PE2 to 11.0.0.102
set protocols mpls label-switched-path LSP_PE4_PE2 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE2 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE2 adaptive
set protocols mpls label-switched-path LSP_PE4_PE3 to 11.0.0.103
set protocols mpls label-switched-path LSP_PE4_PE3 bandwidth 500m
set protocols mpls label-switched-path LSP_PE4_PE3 priority 4 4
set protocols mpls label-switched-path LSP_PE4_PE3 adaptive
set protocols mpls interface all
set protocols mpls interface ge-0/1/9.0 srlg srlg-407
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols pcep pce jnc local-address 11.0.0.104
set protocols pcep pce jnc destination-ipv4-address 192.168.10.100

```

P5

```

set version 14.2X1.1
set system host-name P5
set logical-systems p106 interfaces ge-0/0/3 unit 0 family inet address 11.105.106.2/30
set logical-systems p106 interfaces ge-0/0/3 unit 0 family mpls
set logical-systems p106 interfaces ge-0/0/6 unit 0 family inet address 11.106.107.1/30
set logical-systems p106 interfaces ge-0/0/6 unit 0 family mpls
set logical-systems p106 interfaces ge-0/1/3 unit 0 family inet address 11.102.106.2/30
set logical-systems p106 interfaces ge-0/1/3 unit 0 family mpls
set logical-systems p106 interfaces ge-0/1/7 unit 0 family inet address 11.104.106.2/30
set logical-systems p106 interfaces ge-0/1/7 unit 0 family mpls
set logical-systems p106 interfaces lo0 unit 106 family inet address 11.0.0.106/32
set logical-systems p106 interfaces lo0 unit 106 family mpls
set logical-systems p106 protocols rsvp interface all bandwidth 10g
set logical-systems p106 protocols rsvp interface fxp0.0 disable
set logical-systems p106 protocols rsvp interface ge-0/1/3.0 bandwidth 40g
set logical-systems p106 protocols rsvp interface ge-0/1/7.0 bandwidth 40g
set logical-systems p106 protocols mpls traffic-engineering database import policy TE
set logical-systems p106 protocols mpls interface all
set logical-systems p106 protocols mpls interface fxp0.0 disable
set logical-systems p106 protocols bgp group northstar type internal
set logical-systems p106 protocols bgp group northstar local-address 11.0.0.106
set logical-systems p106 protocols bgp group northstar family traffic-engineering unicast
set logical-systems p106 protocols bgp group northstar export TE
set logical-systems p106 protocols bgp group northstar neighbor 192.168.10.101
set logical-systems p106 protocols ospf traffic-engineering
set logical-systems p106 protocols ospf reference-bandwidth 100g
set logical-systems p106 protocols ospf area 0.0.0.0 interface all interface-type p2p
set logical-systems p106 policy-options policy-statement TE from family traffic-engineering
set logical-systems p106 policy-options policy-statement TE then accept
set logical-systems p106 routing-options srlg srlg-100 srlg-value 100
set logical-systems p106 routing-options srlg srlg-100 srlg-cost 50
set logical-systems p106 routing-options autonomous-system 100
set logical-systems p107 interfaces ge-0/0/5 unit 0 family inet address 11.105.107.2/30
set logical-systems p107 interfaces ge-0/0/5 unit 0 family mpls
set logical-systems p107 interfaces ge-0/0/7 unit 0 family inet address 11.106.107.2/30
set logical-systems p107 interfaces ge-0/0/7 unit 0 family mpls
set logical-systems p107 interfaces ge-0/1/8 unit 0 family inet address 11.103.107.2/30
set logical-systems p107 interfaces ge-0/1/8 unit 0 family mpls
set logical-systems p107 interfaces ge-0/1/9 unit 0 family inet address 11.104.107.2/30
set logical-systems p107 interfaces ge-0/1/9 unit 0 family mpls
set logical-systems p107 interfaces lo0 unit 107 family inet address 11.0.0.107/32
set logical-systems p107 interfaces lo0 unit 107 family mpls
set logical-systems p107 protocols rsvp interface all bandwidth 10g
set logical-systems p107 protocols rsvp interface fxp0.0 disable
set logical-systems p107 protocols rsvp interface ge-0/1/8.0 bandwidth 40g
set logical-systems p107 protocols mpls traffic-engineering database import policy TE
set logical-systems p107 protocols mpls interface all
set logical-systems p107 protocols mpls interface fxp0.0 disable
set logical-systems p107 protocols mpls interface ge-0/0/9.0 srlg srlg-407
set logical-systems p107 protocols mpls interface ge-0/1/9.0 srlg srlg-407
set logical-systems p107 protocols bgp group Northstar type internal
set logical-systems p107 protocols bgp group Northstar local-address 11.0.0.107

```

```
set logical-systems p107 protocols bgp group Northstar family traffic-engineering unicast
set logical-systems p107 protocols bgp group Northstar export TE
set logical-systems p107 protocols bgp group Northstar local-as 100
set logical-systems p107 protocols bgp group Northstar neighbor 192.168.10.101
set logical-systems p107 protocols ospf traffic-engineering
set logical-systems p107 protocols ospf reference-bandwidth 100g
set logical-systems p107 protocols ospf area 0.0.0.0 interface all interface-type p2p
set logical-systems p107 protocols ospf area 0.0.0.1 interface ge-0/1/8.0 interface-type p2p
set logical-systems p107 policy-options policy-statement TE from family traffic-engineering
set logical-systems p107 policy-options policy-statement TE then accept
set logical-systems p107 routing-options srlg srlg-100 srlg-value 100
set logical-systems p107 routing-options srlg srlg-100 srlg-cost 50
set logical-systems p107 routing-options srlg srlg-407 srlg-value 407
set logical-systems p107 routing-options srlg srlg-407 srlg-cost 50
set interfaces ge-0/0/2 unit 0 family inet address 11.105.106.1/30
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/4 unit 0 family inet address 11.105.107.1/30
set interfaces ge-0/0/4 unit 0 family mpls
set interfaces ge-0/0/8 unit 0 family inet address 192.168.10.1/24
set interfaces ge-0/1/1 unit 0 family inet address 11.101.105.2/30
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces ge-0/1/2 unit 0 family inet address 11.102.105.2/30
set interfaces ge-0/1/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 11.0.0.105/32
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg-100 srlg-value 100
set routing-options srlg srlg-100 srlg-cost 50
set routing-options srlg srlg-407 srlg-value 407
set routing-options srlg srlg-407 srlg-cost 50
set routing-options static route 10.161.0.0/16 next-hop 10.92.63.254
set routing-options router-id 11.0.0.105
set routing-options autonomous-system 100
set protocols rsvp interface all bandwidth 10g
set protocols rsvp interface fxp0.0 disable
set protocols rsvp interface ge-0/1/1.0 bandwidth 40g
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf traffic-engineering
set protocols ospf reference-bandwidth 100g
set protocols ospf area 0.0.0.0 interface all interface-type p2p
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface ge-0/0/8.0 passive
```

NorthStar Controller Resources

- The Juniper Networks landing page for the NorthStar Controller: <http://www.juniper.net/us/en/products-services/sdn/northstar-network-controller/>.
- The NorthStar Controller datasheet: <http://www.juniper.net/assets/us/en/local/pdf/datasheets/1000494-en.pdf>.
- *The NorthStar Controller Getting Started Guide*: http://www.juniper.net/techpubs/en_US/northstar1.0.0/information-products/pathway-pages/getting-started.html.
- *The NorthStar Controller User Interface Guide*: http://www.juniper.net/techpubs/en_US/northstar1.0.0/information-products/pathway-pages/northstar-graphical-user-interface.html.

