

DAY ONE: INSIDE SEGMENT ROUTING



By Anurag Khare and Colby Barth

DAY ONE: INSIDE SEGMENT ROUTING

Introducing segment routing (SR) into an existing MPLS network of any worth is a sizable undertaking, but it can be well worth the effort. Far from simply reigniting the debate about how networks should route, segment routing ushers in novel approaches to long-standing practices of node identification, network partitioning, and multipathing. In this book, you'll get inside segment routing with a series of advanced use cases that detail Junos® implementation and deployment solutions with sage production advice from two senior Juniper engineers. Leap in.

"Segment routing moves state from the network and puts it in the packet. Anurag and Colby deal with the intricacies of leveraging segment routing in existing LDP and RSVP-TE networks. Their book provides the keys to deploying and successfully enabling segment routing with a traffic engineering controller, using all of their first-hand experience from initial deployments."

A.E. Natarajan, SVP Junos Engineering, Juniper Networks

"A must read for anyone exploring how to deploy segment routing in their network. You'll get answers to the commonly asked questions – how to migrate to SR, how is the existing network impacted, how to operate a SR-based network, and much more..."

Aman Kapoor, Principal Engineering Manager, Azure Networking, Microsoft

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

- Implement solutions to a variety of issues SR can solve: both IPv4 and native IPv6 connectivity, traffic protection, traffic engineering for constraints-based routing, and bandwidth optimization.
- Plan controlled SR deployments in brownfield networks and migrate away from LDP and/or RSVP-TE.
- Appreciate the importance of modern telemetry stacks, how they improve upon legacy protocols such as SNMP, and their relevance to how bandwidth optimization can be performed.
- Understand the role centralized traffic controllers can play in complex networks, with their ability to visualize the topology, as well as direct both the underlay and overlay.
- Explore cutting-edge tools such as SR-over-UDP and FlexAlgo.



Juniper Networks Books are focused on network reliability.
Peruse the complete library at www.juniper.net/books.

JUNIPER
NETWORKS

Day One: Inside Segment Routing

by Anurag Khare and Colby Barth

<i>Preface</i>	viii
<i>Chapter 1: Connectivity</i>	9
<i>Chapter 2: Interoperability</i>	28
<i>Chapter 3: Observability</i>	57
<i>Chapter 4: Optimizability</i>	80
<i>Chapter 5: Extensibility</i>	111
<i>Appendix</i>	121

© 2019 by Juniper Networks, Inc.

All rights reserved. Juniper Networks and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo and the Junos logo, 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: Anurag Khare, Colby Barth

Technical Reviewers: Harish Sitaraman, Shraddha Hegde, Peter Van Oene

Editor in Chief: Patrick Ames

Copyeditor: Nancy Koerbel

ISBN: 978-1-941441-95-4 (print)

Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-94-7 (ebook)

Version History: v1, May 2019

2 3 4 5 6 7 8 9 10

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

About the Authors

Anurag Khare is a Principal Engineer at Juniper Networks. He has been building, breaking, then re-jiggering every kind of network he can get his hands on for over 17 years.

Colby Barth is a Distinguished Engineer at Juniper Networks. Colby has over 20 years of experience, designing, building, and operating IP/MPLS networks.

Authors' Acknowledgments

We thank Shraddha Hegde, William Britto AJ, Upendra Singh, and Chris Bowers for providing access to features in development; Richard Chen, Nur Hanita Binti Mohd-Murad, and Patricio Giecco for helping troubleshoot our text, and finally, our customers for their vast interest in this technology.

Feedback? Comments? Error reports? Email them to dayone@juniper.net.

Welcome to Day One

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

Day One books cover the Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow. You can obtain the books from various sources:

- Download a free PDF edition at <http://www.juniper.net/dayone>.
- Many of the library's books are available on the Juniper app: [Junos Genius](#).
- Get the ebook edition for iPhones and iPads from the iBooks Store. Search for *Juniper Networks Books* or the title of this book.
- 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 Amazon Kindle Store. Search for *Juniper Networks Books* or the title of this book.
- Purchase the paper edition at Vervante Corporation (www.vervante.com) for between \$15-\$40, depending on page length.
- Note that most mobile devices can also view PDF files.

Key Segment Routing Resources

Segment routing (SR) is a strategic end-to-end transport architecture. There are some key resources you should be familiar with before starting this book:

- The Juniper TechLibrary is full of SR information for Junos and for individual hardware platforms. This book is not a substitute for that body of work, so you should review the documentation: https://www.juniper.net/documentation/en_US/junos/topics/topic-map/static-segment-routing-lsp.html#jd0e33.
- *Day One: Configuring Segment Routing with Junos* will guide you in setting up SR in your lab and production environment. This book assumes you have read and understand these SR concepts and initial configurations. It's a free PDF and its authors are active on the J-Net Forums. Please review here: <https://www.juniper.net/us/en/training/jnbooks/day-one/configuring-segment-routing-junos/index.page>.
- Catch a quick video on YouTube by Juniper CTO Bikash Koley on Juniper and the future of SR: https://www.youtube.com/watch?v=MA79_8tqL_I.

What You Need to Know Before Reading This Book

The authors have made a few assumptions about the general knowledge level of the reader:

- You have read *Day One: Configuring Segment Routing with Junos* as mentioned on the previous page.
- You understand how existing Multiprotocol Label Switching (MPLS) networks are designed, configured, and how they function. This includes a basic understanding of IGPs, BGP, LDP, and RSVP-TE.
- You understand how traffic engineering improves resource utilization in current wide-area networks.

NOTE There are several books in the *Day One* library covering MPLS, BGP, and the Junos CLI for various levels of expertise: <http://www.juniper.net/dayone>.

What You Will Learn by Reading This Book

This book will help you to:

- Successfully understand and implement solutions to the variety of problems segment routing (SR) solves: both IPv4 and native IPv6 connectivity, traffic protection, traffic engineering for constraints-based routing, and bandwidth optimization.
- Plan controlled SR deployments in brownfield networks and migrate away from LDP and/or RSVP-TE. This book accretes SR functionality to a typical service provider wide-area network, gradually replacing the existing with equivalent, and in most cases superior, functionality.
- Appreciate the importance of modern telemetry stacks, how they improve upon legacy protocols such as SNMP, and their relevance to how bandwidth optimization can be performed.
- Understand the role centralized traffic controllers can play in complex networks, with their ability to visualize the topology, as well as direct both the underlay and overlay.
- Explore cutting-edge tools such as SR-over-UDP and FlexAlgo. SRoUDP is gaining relevance as a viable data plane for non-SR-MPLS capable platforms. FlexAlgo is a fresh look at multi-topology routing, without the limitations of earlier approaches.

Glossary of Used Terms

BGP	Border Gateway Protocol
BGP-LU	Border Gateway Protocol Labeled Unicast
CoS	Class of Service
ECMP	Equal Cost Multipath
EVPN	Ethernet VPN
FA	Forwarding Adjacency
IS-IS	Intermediate System-to-Intermediate System
LDP	Label Distribution Protocol
LSDB	Link State Database
LSP	Label Switched Path
MPLS	Multiprotocol Label Switching
OSPF	Open Shortest Path First
PHP	Penultimate Hop Pop
RSVP-TE	Resource Reservation Protocol - Traffic Engineering
TED	Traffic Engineering Database
UHP	Ultimate Hop Pop
VPN	Virtual Private Network

Preface

For decades multiprotocol label switched (MPLS) networks have been pushing, popping, and swapping countless labels on untold packets. Labels still remain the fundamental MPLS forwarding instruction. A label-switched path (LSP) involves a contiguous set of routers where, at the simplest level, each performs one or more of the same three forwarding actions —a push, pop, or swap.

An LSP can be handcrafted, with each router's management plane configured with the appropriate action. More commonly, control plane signaling based on LDP, RSVP-TE, or BGP-LU, weaves an LSP. Neither protocol is mutually exclusive and some of the most demanding networks use a combination.

Segment routing (SR) introduces a fourth option to this trio of well-known label distribution protocols and it does this without resorting to specifying yet another protocol. SR prefers instead to overload existing interior and exterior gateway protocols – IS-IS, OSPF, and BGP. Their protocol machinery is reused with a combination of new attributes that disseminate information about segments, SR's atomic-like abstraction of a forwarding instruction.

By using segments, SR decouples itself from MPLS in order to be forwarding plane agnostic. SR-over-UDP uses the well-treaded IPv4 forwarding plane and is under active development. SRv6 and SRv6+, whose forwarding plane is IPv6, are being deliberated over by standards bodies. That said, SR with an MPLS forwarding plane remains in its most approachable form. The familiar push, pop, and swap actions are referred to as push, next, and continue, respectively, in SR parlance.

In addition to forwarding plane independence, SR proposes novel and attractive approaches to traffic protection and engineering. Consequently, it delivers both the equal-cost multipathing (ECMP) capabilities of IP routing, along with the ability to precisely engineer a path à la RSVP-TE.

In spite of these desirable qualities, introducing SR into an existing MPLS network of any worth is a sizeable undertaking. Multiple label distribution protocols will be active in the network – at least temporarily – and the timespan is likely to be more in the order of months, not weeks, or days. On some routers, protocols will overlap and a configuration will be needed to indicate which is preferred.

The undertaking is well worth the effort. Far from simply reigniting the debate about how networks should route, segment routing ushers in novel approaches to long-standing practices of node identification, network partitioning, and multipathing.

MORE? For the reader entirely unfamiliar with SR, the first few pages of Chapter 1 provide an extremely compressed introduction. For a complete introduction, read the SR companion book, *Day One: Configuring Segment Routing with Junos*, by Lucek and Szarkowicz, at <https://www.juniper.net/us/en/training/jnbooks/day-one/configuring-segment-routing-junos/index.page>.

Chapter 1

Connectivity

This chapter introduces a *brownfield* network (contains legacy systems) that uses LDP at the edge, and that tunnels over an RSVP-TE-only core. The chapter's first task is to introduce SR intelligently and deliberately; then it will look at the extent to which SR can subsume all label distribution responsibility.

NOTE Some of the functionality detailed here may not yet have become generally available by the time this book is published. Please contact your Juniper account manager for timelines.

Reachability

Essential connectivity in SR is achieved by advertising *prefix* and *adjacency segments*. These are denoted by *segment identifiers* (SID). Many other types of SIDs exist, each serving a different purpose, but these two suffice to establish basic reachability. The prefix, and its specialized node SID form, uniquely identifies a router in a given topology, reachable through a particular path-finding algorithm; the adjacency SID identifies a routed link to an IGP neighbor.

NOTE There are more segment types than there is room to discuss them. Anycast segments represent a shared prefix across nodes that could be used for simple, stateless load-sharing and redundancy. A Level 2 adjacency SID represents individual member links of a LAG bundle and is expected to be used to verify per-member-link OAM; its corollary is the adjacency set that bundles adjacencies into a single segment. Binding SIDs represent tunnels. Some of these SIDs are explored in detail in *Chapter 3: Optimizability*.

Some segments are directly advertised as labels, as in the case of adjacency SIDs; prefix and node SIDs are indirectly advertised as indexes into an SR Global Block (SRGB). The SRGB represents the label space that nodes use to refer to global segments. It is one reason SR does not need to advertise a label for each IGP-learned prefix. The SRGB is configurable, and ideally consistent across the SR domain.

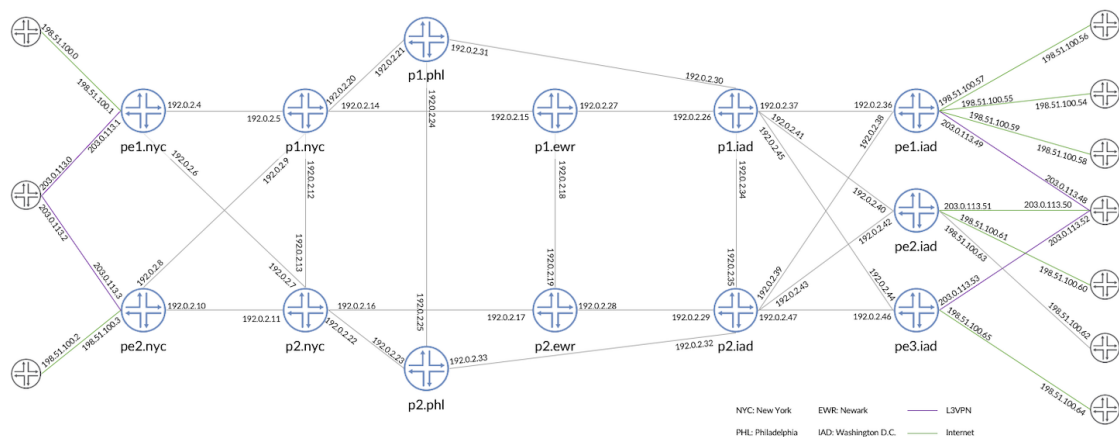
There is at least one segment associated with an IPv4 prefix and possibly another for IPv6. Node segments are associated with the router's loopback address. Indeed, the IPv4 and IPv6 node indexes are expected to be treated just as loopback address are – managed by the operator and unique per-node in an SR domain.

Adjacency segments default to being dynamically assigned labels with a pop, and forwarded via the corresponding IGP adjacency action. These are used for strict steering, including traffic protection. Unlike node SIDs, adjacency SIDs have the option to be locally or globally significant (albeit, globally significant adjacency SIDs are uncommon).

NOTE Strictly speaking, node SIDs are only globally consistent when all routers share a common SRGB. This is the recommended deployment mode. A globally consistent node SID greatly simplifies operations, including troubleshooting, as a given router is represented by an identical entry in the label forwarding information base (LFIB) of all other routers in the same SR domain.

Brownfields

Our example network is depicted in Figure 1.1. This is a common design for many carriers. Both Internet and VPN overlay service are delivered by the combination of LDP/RSVP-TE as label distribution protocols, and multi-area IS-IS as the IGP of choice.



Since our focus is the underlay, let Internet and L3VPN connectivity suffice as proof of our work. Of course, any MPLS service – 6PE, 6VPE, L2VPN, EVPN, et. al., – could be delivered over this same network.

To orient ourselves, let's run a traceroute from the CE in New York (ce1.nyc) to Washington (ce1.iad).

NOTE For the sake of brevity, Washington, DC will be called Washington from now on.

Connectivity verification

```
user@ce1.nyc> traceroute 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1)  2.427 ms  1.646 ms  3.676 ms
 2 p2.nyc-ge-0-0-2.0 (192.0.2.7)  70.034 ms p1.nyc-ge-0-0-2.0 (192.0.2.5)  7.387 ms p2.nyc-
ge-0-0-2.0 (192.0.2.7)  52.897 ms
   MPLS Label=119 CoS=0 TTL=1 S=1
 3 p1.phl-ge-0-0-2.0 (192.0.2.21)  55.576 ms p2.ewr-ge-0-0-2.0 (192.0.2.17)  6.849 ms p1.phl-
ge-0-0-2.0 (192.0.2.21)  80.628 ms
   MPLS Label=313824 CoS=0 TTL=1 S=0
   MPLS Label=352896 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-6.0 (192.0.2.28)  7.129 ms p1.iad-ge-0-0-5.0 (192.0.2.30)  80.143 ms  73.898 ms
   MPLS Label=352896 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36)  6.994 ms  8.681 ms  5.971 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54)  8.125 ms  7.631 ms  10.807 ms
```

Note that the labels in use will change once SR starts to be used.

For now, let's enable SR on pe1.nyc and carefully trace the changes to the control and forwarding planes.

Enabling SR on pe1.nyc

```
user@pe1.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 1;

```

The SRGB is configured to range from 1000 to 1127, inclusive. Within that range, this router represents its IPv4 loopback address as 1001 (1000 + 1, its IPv4-index). Let's confirm this to be the case.

Control plane: node SID verification

```
user@pe1.nyc> show isis overview
Instance: master
  Router ID: 128.49.106.1
  Hostname: pe1.nyc
```

```

Sysid: 0128.0049.1061
Areaid: 49.0001
Adjacency holddown: enabled
Maximum Areas: 3
LSP life time: 1200
Attached bit evaluation: enabled
SPF delay: 200 msec, SPF holddown: 5000 msec, SPF rapid runs: 3
Overload bit at startup is set
  Overload high metrics: disabled
  Allow route leaking: disabled
  Allow internal prefix overloading: disabled
  Allow external prefix overloading: disabled
IPv4 is enabled, IPv6 is enabled, SPRING based MPLS is enabled
Traffic engineering: enabled
Restart: Disabled
  Helper mode: Enabled
Layer2-map: Disabled
Source Packet Routing (SPRING): Enabled
  SRGB Config Range:
    SRGB Start-Label : 1000, SRGB Index-Range : 128
    SRGB Block Allocation: Success
    SRGB Start Index : 1000, SRGB Size : 128, Label-Range: [ 1000, 1127 ]
  Node Segments: Enabled
    Ipv4 Index : 1, Ipv6 Index : 61
Post Convergence Backup: Disabled
Level 1
  Internal route preference: 15
  External route preference: 160
  Prefix export count: 0
  Wide metrics are enabled
  Source Packet Routing is enabled
Level 2
  Internal route preference: 18
  External route preference: 165
  Prefix export count: 0
  Wide metrics are enabled, Narrow metrics are enabled
  Source Packet Routing is enabled

```

Apart from the manually-configured node SID, let's verify that adjacency SIDs have also been dynamically allocated.

Control plane: adjacency SID verification

```

user@pe1.nyc> show isis database pe1.nyc level 1 extensive
IS-IS level 1 link-state database:

```

```

pe1.nyc.00-00 Sequence: 0x496, Checksum: 0x981e, Lifetime: 771 secs
  IPV4 Index: 1
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
  IS neighbor: p2.nyc.00 Metric: 10
    Two-way fragment: p2.nyc.00-00, Two-way first fragment: p2.nyc.00-00
    P2P IPv4 Adj-SID: 26, Weight: 0, Flags: --VL--
  IS neighbor: p1.nyc.00 Metric: 10

```

```

    Two-way fragment: p1.nyc.00-00, Two-way first fragment: p1.nyc.00-00
    P2P IPv4 Adj-SID:      27, Weight:  0, Flags: --VL--
IP prefix: 128.49.106.1/32      Metric:      0 Internal Up

Header: LSP ID: pe1.nyc.00-00, Length: 191 bytes
Allocated length: 1492 bytes, Router ID: 128.49.106.1
Remaining lifetime: 771 secs, Level: 1, Interface: 0
Estimated free bytes: 1257, Actual free bytes: 1301
Aging timer expires in: 771 secs
Protocols: IP, IPv6

Packet: LSP ID: pe1.nyc.00-00, Length: 191 bytes, Lifetime : 1198 secs
Checksum: 0x981e, Sequence: 0x496, Attributes: 0x5 <L1 Overload>
NLPID: 0x83, Fixed length: 27 bytes, Version: 1, Sysid length: 0 bytes
Packet type: 18, Packet version: 1, Max area: 0

TLVs:
Area address: 49.0001 (3)
LSP Buffer Size: 1492
Speaks: IP
Speaks: IPv6
IP router id: 128.49.106.1
IP address: 128.49.106.1
Hostname: pe1.nyc
Router Capability: Router ID 128.49.106.1, Flags: 0x00
  SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 128, SID-Label: 1000
  SPRING Algorithm - Algo: 0
Extended IS Reachability TLV, Type: 22, Length: 80
IS extended neighbor: p2.nyc.00, Metric: default 10 SubTLV len: 29
  IP address: 192.0.2.6
  Neighbor's IP address: 192.0.2.7
  Local interface index: 336, Remote interface index: 334
  P2P IPV4 Adj-SID - Flags: 0x30(F:0,B:0,V:1,L:1,S:0,P:0), Weight:0, Label: 26
  P2P IPv4 Adj-SID:      26, Weight:  0, Flags: --VL--
IS extended neighbor: p1.nyc.00, Metric: default 10 SubTLV len: 29
  IP address: 192.0.2.4
  Neighbor's IP address: 192.0.2.5
  Local interface index: 335, Remote interface index: 334
  P2P IPV4 Adj-SID - Flags: 0x30(F:0,B:0,V:1,L:1,S:0,P:0), Weight:0, Label: 27
  P2P IPv4 Adj-SID:      27, Weight:  0, Flags: --VL--
IP extended prefix: 128.49.106.1/32 metric 0 up
  14 bytes of subtlvs
  Administrative tag 1: 1111
  Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 1
No queued transmissions

```

There is one adjacency SID to each of pe1.nyc's IS-IS neighbors. Additionally, the node SID is highlighted as a sub-TLV of the IP-extended prefix TLV. The flags also clarify and contrast between the two SID types. 'V' indicates whether the subTLV carries a value (as in the case of an adjacency SID) or an index (as with a node SID). 'L' represents local vs. global significance. Finally, the 'N' flag states that a given prefix SID is more specifically a node SID as it is attached to the router's loopback address.

Forwarding plane: PE1's FIB

```
user@pe1.nyc> show route table mpls.0 protocol isis
```

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

```
26          *[L-ISIS/14] 00:44:02, metric 0
              > to 192.0.2.7 via ge-0/0/2.0, Pop
26(S=0)     *[L-ISIS/14] 00:02:40, metric 0
              > to 192.0.2.7 via ge-0/0/2.0, Pop
27          *[L-ISIS/14] 00:44:02, metric 0
              > to 192.0.2.5 via ge-0/0/1.0, Pop
27(S=0)     *[L-ISIS/14] 00:02:40, metric 0
              > to 192.0.2.5 via ge-0/0/1.0, Pop
```

L-ISIS stands for labeled IS-IS. The adjacency SIDs are installed in the FIB but the node SID seems to be missing. This is normal. Since the ‘P’ flag (PHP-off) was not set, this router does not expect to receive packets with the top label 1001. Instead, 1001 would be popped by its neighbors (once they are enabled for SR). This is analogous to implicit-null behavior used by traditional label distribution protocols.

Inter-area

Since none of the other routers are currently enabled for SR, pe1.nyc's SIDs aren't yet learned by area-external routers. Intra-area routers, such as pe2.nyc, learn but ignore them.

Connectivity verification

```
user@ce1.nyc> traceroute 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1) 6.659 ms 2.265 ms 2.116 ms
 2 p2.nyc-ge-0-0-2.0 (192.0.2.7) 8.626 ms 8.556 ms 13.895 ms
    MPLS Label=119 CoS=0 TTL=1 S=1
 3 p2.ewr-ge-0-0-2.0 (192.0.2.17) 20.193 ms p1.phl-ge-0-0-2.0 (192.0.2.21) 18.045 ms 11.514 ms
    MPLS Label=313824 CoS=0 TTL=1 S=0
    MPLS Label=352896 CoS=0 TTL=1 S=1
 4 p1.iad-ge-0-0-5.0 (192.0.2.30) 10.881 ms 9.305 ms 8.683 ms
    MPLS Label=352896 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 8.032 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 8.232 ms pe1.
iad-ge-0-0-1.0 (192.0.2.36) 7.529 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 15.007 ms 10.747 ms 12.320 ms
```

Nothing has changed, as one would expect. Labeled IS-IS has a lower protocol preference than LDP or RSVP-TE, and requires operator intervention to start being used. No SR forwarding is taking place as our SR domain is an island of one, as shown in Figure 1.2. In the next section, we'll configure node SIDs on all routers in New York (which also share a common IS-IS area), starting with the L1-L2 attached p1.nyc.


```

    Two-way fragment: p2.nyc.00-00, Two-way first fragment: p2.nyc.00-00
    P2P IPv4 Adj-SID:      32, Weight:  0, Flags: --VL--
...
  TLVs:
...
    IP extended prefix: 128.49.106.3/32 metric 0 up
    14 bytes of subtlvs
    Administrative tag 1: 1111
    Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 3

IS-IS level 2 link-state database:

p1.nyc.00-00 Sequence: 0x758, Checksum: 0xc13, Lifetime: 967 secs
  IPv4 Index: 3
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
  IS neighbor: p2.nyc.00 Metric: 999999
    Two-way fragment: p2.nyc.00-00, Two-way first fragment: p2.nyc.00-00
    P2P IPv4 Adj-SID:      33, Weight:  0, Flags: --VL--
  IS neighbor: p1.ewr.00 Metric: 10
    Two-way fragment: p1.ewr.00-00, Two-way first fragment: p1.ewr.00-00
    P2P IPv4 Adj-SID:      29, Weight:  0, Flags: --VL--
  IS neighbor: p1.phl.00 Metric: 10
    Two-way fragment: p1.phl.00-00, Two-way first fragment: p1.phl.00-00
    P2P IPv4 Adj-SID:      28, Weight:  0, Flags: --VL--
...
  TLVs:
...
    IP extended prefix: 128.49.106.3/32 metric 0 up
    14 bytes of subtlvs
    Administrative tag 1: 1111
    Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 3
    IP extended prefix: 128.49.106.1/32 metric 10 up
    14 bytes of subtlvs
    Administrative tag 1: 1111
    Node SID, Flags: 0xe0(R:1,N:1,P:1,E:0,V:0,L:0), Algo: SPF(0), Value: 1
...

```

Two differences are apparent compared to pe1.nyc. As an area border router (ABR), p1.nyc not only advertises its own node SID into both the L1 and L2 areas, it also re-originates pe1.nyc's node SID advertisement across areas. This allows global segments to be learned across IGP areas. The 'R' flag indicates this node SID is being re-advertised across areas/levels.

Crucially, the 'P' flag is set on this re-advertised node SID, in stark contrast to p1.nyc's own node SID. This signals to SR-capable neighbors that p1.nyc expects them to use the continue (swap) action and retain the top label which it interprets as a forwarding instruction to pe1.nyc.

Forwarding plane: P1's FIB

We can confirm that p1.nyc has installed an entry for label 1001, which represents pe1.nyc's node SID:

```

user@pe1.nyc> show route table mpls.0 protocol isis

mpls.0: 31 destinations, 31 routes (31 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

28          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.21 via ge-0/0/5.0, Pop
28(S=0)     *[L-ISIS/14] 00:09:21, metric 0
             > to 192.0.2.21 via ge-0/0/5.0, Pop
29          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.15 via ge-0/0/4.0, Pop
29(S=0)     *[L-ISIS/14] 00:09:21, metric 0
             > to 192.0.2.15 via ge-0/0/4.0, Pop
30          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.8 via ge-0/0/3.0, Pop
30(S=0)     *[L-ISIS/14] 00:07:10, metric 0
             > to 192.0.2.8 via ge-0/0/3.0, Pop
31          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.4 via ge-0/0/2.0, Pop
31(S=0)     *[L-ISIS/14] 00:07:10, metric 0
             > to 192.0.2.4 via ge-0/0/2.0, Pop
32          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.13 via ge-0/0/1.0, Pop
32(S=0)     *[L-ISIS/14] 00:07:10, metric 0
             > to 192.0.2.13 via ge-0/0/1.0, Pop
33          *[L-ISIS/14] 00:34:55, metric 0
             > to 192.0.2.13 via ge-0/0/1.0, Pop
33(S=0)     *[L-ISIS/14] 00:09:21, metric 0
             > to 192.0.2.13 via ge-0/0/1.0, Pop
1001        *[L-ISIS/14] 00:34:55, metric 10
             > to 192.0.2.4 via ge-0/0/2.0, Pop
1001(S=0)   *[L-ISIS/14] 00:07:10, metric 10
             > to 192.0.2.4 via ge-0/0/2.0, Pop

```

Not only can you see a pop (the MPLS forwarding plane's version of *continue*) action for incoming label 1001, you can see pop actions for adjacency SID labels 28-33.

Forwarding plane: PE1's FIB

```

user@pe1.nyc> show route protocol isis table mpls.0

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

26          *[L-ISIS/14] 02:06:55, metric 0
             > to 192.0.2.7 via ge-0/0/2.0, Pop
26(S=0)     *[L-ISIS/14] 00:10:50, metric 0
             > to 192.0.2.7 via ge-0/0/2.0, Pop
27          *[L-ISIS/14] 00:48:12, metric 0
             > to 192.0.2.5 via ge-0/0/1.0, Pop
27(S=0)     *[L-ISIS/14] 00:10:50, metric 0
             > to 192.0.2.5 via ge-0/0/1.0, Pop
1003        *[L-ISIS/14] 00:35:14, metric 10
             > to 192.0.2.5 via ge-0/0/1.0, Pop
1003(S=0)   *[L-ISIS/14] 00:10:50, metric 10
             > to 192.0.2.5 via ge-0/0/1.0, Pop

```

On pe1.nyc there is now a forwarding entry for label 1003, representing p1.nyc's node SID.

Inter-area

Other than pe1.nyc, none of p1.nyc's other neighbors are SR-enabled. All L2 neighbors learn about, but ignore, the segments.

Connectivity verification

```
user@ce1.nyc> traceroute 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1) 3.118 ms 2.042 ms 1.803 ms
 2 p2.nyc-ge-0-0-2.0 (192.0.2.7) 7.747 ms p1.nyc-ge-0-0-2.0 (192.0.2.5) 101.823 ms p2.nyc-
ge-0-0-2.0 (192.0.2.7) 23.312 ms
   MPLS Label=119 CoS=0 TTL=1 S=1
 3 p1.phl-ge-0-0-2.0 (192.0.2.21) 7.152 ms p2.ewr-ge-0-0-2.0 (192.0.2.17) 69.566 ms 67.497 ms
   MPLS Label=314 CoS=0 TTL=1 S=0
   MPLS Label=355776 CoS=0 TTL=1 S=1
 4 p1.iad-ge-0-0-5.0 (192.0.2.30) 11.528 ms 9.755 ms p2.iad-ge-0-0-6.0 (192.0.2.28) 6.276 ms
   MPLS Label=355776 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 5.667 ms 8.056 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 10.915 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 169.285 ms 11.048 ms 9.896 ms
```

As unexciting as the lack of change in forwarding behavior may be, this is reassuring. Enabling SR has not inadvertently changed, let alone harmed, existing services. After all, a controlled deployment is preferred.

Enabling SR on the remaining New York routers

```
user@pe2.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 0;
      }
    }
  }
}
```

```
user@p2.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 2;
      }
    }
  }
}
```

Control plane

At this point you should know what to expect. Two new node SIDs would be advertised into the L1 area; pe2.nyc's node SID would be re-advertised by both p1.nyc and p2.nyc into the L2 subdomain. Like pe1.nyc's before it, it would be ignored by the non-SR enabled L2 routers for now. Let's see:

```
user@p2.nyc> show isis database p2.nyc extensive
```

```
IS-IS level 1 link-state database:
```

```
p2.nyc.00-00 Sequence: 0x4a3, Checksum: 0x28ab, Lifetime: 851 secs
```

```
IPV4 Index: 2
```

```
Node Segment Blocks Advertised:
```

```
Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
```

```
IS neighbor: pe2.nyc.00 Metric: 10
```

```
Two-way fragment: pe2.nyc.00-00, Two-way first fragment: pe2.nyc.00-00
```

```
P2P IPv4 Adj-SID: 265, Weight: 0, Flags: --VL--
```

```
IS neighbor: pe1.nyc.00 Metric: 10
```

```
Two-way fragment: pe1.nyc.00-00, Two-way first fragment: pe1.nyc.00-00
```

```
P2P IPv4 Adj-SID: 266, Weight: 0, Flags: --VL--
```

```
IS neighbor: p1.nyc.00 Metric: 10
```

```
Two-way fragment: p1.nyc.00-00, Two-way first fragment: p1.nyc.00-00
```

```
P2P IPv4 Adj-SID: 267, Weight: 0, Flags: --VL--
```

```
...
```

```
TLVs:
```

```
...
```

```
IP extended prefix: 128.49.106.2/32 metric 0 up
```

```
14 bytes of subtlvs
```

```
Administrative tag 1: 1111
```

```
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 2
```

```
...
```

```
IS-IS level 2 link-state database:
```

```
p2.nyc.00-00 Sequence: 0x4a0, Checksum: 0x8f75, Lifetime: 851 secs
```

```
IPV4 Index: 2
```

```
Node Segment Blocks Advertised:
```

```
Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
```

```
IS neighbor: p1.nyc.00 Metric: 999999
```

```
Two-way fragment: p1.nyc.00-00, Two-way first fragment: p1.nyc.00-00
```

```
P2P IPv4 Adj-SID: 268, Weight: 0, Flags: --VL--
```

```
IS neighbor: p2.ewr.00 Metric: 10
```

```
Two-way fragment: p2.ewr.00-00, Two-way first fragment: p2.ewr.00-00
```

```
P2P IPv4 Adj-SID: 264, Weight: 0, Flags: --VL--
```

```
IS neighbor: p2.phl.00 Metric: 10
```

```
Two-way fragment: p2.phl.00-00, Two-way first fragment: p2.phl.00-00
```

```
P2P IPv4 Adj-SID: 263, Weight: 0, Flags: --VL--
```

```
...
```

```
TLVs:
```

```
...
```

```
IP extended prefix: 128.49.106.2/32 metric 0 up
```

```
14 bytes of subtlvs
```

```
Administrative tag 1: 1111
```

```
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 2
```

```
IP extended prefix: 128.49.106.0/32 metric 10 up
```

```
14 bytes of subtlvs
```

```
Administrative tag 1: 1111
```

```
Node SID, Flags: 0xe0(R:1,N:1,P:1,E:0,V:0,L:0), Algo: SPF(0), Value: 0
```

```
IP extended prefix: 128.49.106.1/32 metric 10 up
```

```
14 bytes of subtlvs
```

```
Administrative tag 1: 1111
```

```
Node SID, Flags: 0xe0(R:1,N:1,P:1,E:0,V:0,L:0), Algo: SPF(0), Value: 1
```

```
IP extended prefix: 128.49.106.3/32 metric 10 up
```

```
14 bytes of subtlvs
```

```
Administrative tag 1: 1111
```

```
Node SID, Flags: 0xe0(R:1,N:1,P:1,E:0,V:0,L:0), Algo: SPF(0), Value: 3
```

This should now be familiar, there are adjacency SIDs for each IS-IS neighbor (SR-capable or not), as well as node SIDs (re-advertised in all cases but one) for all SR-capable routers.

Connectivity verification

```
user@ce1.nyc> traceroute 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1) 27.852 ms 38.214 ms 2.399 ms
 2 p2.nyc-ge-0-0-2.0 (192.0.2.7) 8.883 ms p1.nyc-ge-0-0-2.0 (192.0.2.5) 7.776 ms p2.nyc-
ge-0-0-2.0 (192.0.2.7) 6.584 ms
   MPLS Label=119 CoS=0 TTL=1 S=1
 3 p1.phl-ge-0-0-2.0 (192.0.2.21) 6.929 ms p2.ewr-ge-0-0-2.0 (192.0.2.17) 7.092 ms 8.230 ms
   MPLS Label=314 CoS=0 TTL=1 S=0
   MPLS Label=355776 CoS=0 TTL=1 S=1
 4 p1.iad-ge-0-0-5.0 (192.0.2.30) 7.097 ms 6.988 ms p2.iad-ge-0-0-6.0 (192.0.2.28) 7.142 ms
   MPLS Label=355776 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 6.132 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 8.677 ms 5.444 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 11.672 ms 9.005 ms 7.873 ms
```

Finally you can confirm no change to traffic forwarding.

All this work to no end? Hardly. It has been proven that enabling the SR control plane does not force an operator to immediately utilize its forwarding plane. Indeed, this is likely to be the first step in any migration to SR – control plane verification coupled with continued service assurance.

Switching to SR

SR is now enabled throughout New York. All it will take to make the adjustment is a protocol preference on an ingress router. So far we have been testing forwarding between CEs in New York and Washington. Since our SR domain is constrained to the former, let's quickly verify connectivity between CEs within the region.

Connectivity verification

```
user@ce1.nyc> traceroute routing-instance svc-inet 198.51.100.0
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 235.860 ms 226.521 ms 69.491 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 11.960 ms 4.510 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 4.477 ms
   MPLS Label=257 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-1.0 (192.0.2.4) 3.625 ms 4.503 ms pe1.nyc-ge-0-0-2.0 (192.0.2.6) 5.833 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 6.088 ms 6.480 ms 7.580 ms
```

None of the SR labels are in use. This makes sense since pe2.nyc, the ingress router, continues to prefer LDP-learned next hops when doing BGP service route resolution.

Control plane

Let's verify the service route and its next hop on ingress PE:

```

user@pe2.nyc> show route 198.51.100.0 active-path extensive

inet.0: 30 destinations, 37 routes (30 active, 0 holddown, 0 hidden)
198.51.100.0/31 (2 entries, 1 announced)
...
    Indirect next hops: 1
        Protocol next hop: 128.49.106.1 Metric: 1
        Indirect next hop: 0xba57f80 1048577 INH Session ID: 0x165
    Indirect path forwarding next hops: 2
        Next hop type: Router
        Next hop: 192.0.2.9 via ge-0/0/1.0 weight 0x1
        Session Id: 0x0
        Next hop: 192.0.2.11 via ge-0/0/2.0 weight 0x1
        Session Id: 0x0

user@pe2.nyc> show route 128.49.106.1 table inet.3

inet.3: 8 destinations, 11 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.1/32    *[LDP/9] 00:54:50, metric 1
                  to 192.0.2.9 via ge-0/0/1.0, Push 18
                  > to 192.0.2.11 via ge-0/0/2.0, Push 257
                  [L-ISIS/14] 00:54:29, metric 20
                  to 192.0.2.9 via ge-0/0/1.0, Push 1001
                  > to 192.0.2.11 via ge-0/0/2.0, Push 1001

user@pe2.nyc> show isis adjacency
Interface          System          L State          Hold (secs) SNPA
ge-0/0/1.0         p1.nyc          1 Up              26
ge-0/0/2.0         p2.nyc          1 Up              26

user@pe2.nyc> show ldp database | match "put|128.49.106.1/32"
Input label database, 128.49.106.0:0--128.49.106.2:0
    257      128.49.106.1/32
...

```

The BGP route we're tracing is learned with a protocol next hop of 128.49.106.1 – this is pe1.nyc's loopback address. A label is associated with this forwarding equivalence class (FEC) from both LDP and SR (L-ISIS stands for labeled IS-IS) neighbors. The former has a superior protocol preference, leading to the pe2.nyc imposing the LDP-learned label 257 from its chosen neighbor (p2.nyc).

NOTE Since ECMP is in effect, pe2.nyc could equally as well have pushed label 18 which is advertised by p1.nyc for the same FEC. This is a forwarding plane decision. It is a result of hashing incoming packet headers and using the result to select a single outbound path on a per-flow basis.

To make the change, let's improve the protocol preference for L-ISIS on pe2.nyc.

Prefer SR for intra-region traffic originating at pe2.nyc

```
user@pe2.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 0;
      }
    }
    level 1 labeled-preference 8;
  }
}
```

Control plane: confirm pe2.nyc performs route resolution using SR instead of LDP

```
user@pe2.nyc> show route 128.49.106.1 table inet.3

inet.3: 8 destinations, 11 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.1/32    *[L-ISIS/8] 00:00:41, metric 20
                  to 192.0.2.9 via ge-0/0/1.0, Push 1001
                  > to 192.0.2.11 via ge-0/0/2.0, Push 1001
                  [LDP/9] 01:10:36, metric 1
                  to 192.0.2.9 via ge-0/0/1.0, Push 18
                  > to 192.0.2.11 via ge-0/0/2.0, Push 257
```

The protocol next hop now prefers the SR divined label (1001, pe1.nyc's node SID). This should have (finally!) affected the forwarding plane.

Connectivity verification: intra-NY traffic is SR-switched

```
user@ce1.nyc> traceroute routing-instance svc-inet 198.51.100.0
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 2.903 ms 3.001 ms 1.934 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 5.263 ms p1.nyc-ge-0-0-3.0 (192.0.2.9) 35.811 ms 87.604 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-1.0 (192.0.2.4) 188.617 ms 47.428 ms 8.965 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 10.858 ms 5.580 ms 5.275 ms
```

Note the change in the label pushed by pe2.nyc from 257 to 1001. This confirms traffic in the SR domain is now exercising the LSP woven by segments. Just to make sure all extra-regional traffic remains unaffected by the protocol preference change on pe2.nyc, a trace to the cross-country destination in Washington, remains traditionally label-switched.

Connectivity verification: extra-NY traffic still uses LDP

```
user@ce1.nyc> traceroute 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 49.540 ms 78.154 ms 68.680 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 62.733 ms 64.765 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 64.562 ms
   MPLS Label=119 CoS=0 TTL=1 S=1
 3 p1.phl-ge-0-0-2.0 (192.0.2.21) 60.903 ms 6.794 ms p2.ewr-ge-0-0-2.0 (192.0.2.17) 33.531 ms
   MPLS Label=314 CoS=0 TTL=1 S=0
   MPLS Label=355776 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-6.0 (192.0.2.28) 6.553 ms p1.iad-ge-0-0-5.0 (192.0.2.30) 96.063 ms 67.346 ms
   MPLS Label=352896 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-2.0 (192.0.2.38) 8.891 ms pe1.iad-ge-0-0-1.0 (192.0.2.36) 6.837 ms 6.421 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 30.158 ms 108.181 ms 84.979 ms
```

Now that we are convinced there is no discontinuity of service as a result of our actions, let's mirror pe2.nyc's SR preference on all New York routers. The configuration is identical:

```
user@p1.nyc> show configuration
protocols {
  isis {
    ...
    level 1 labeled-preference 8;
```

At this point, the New York routers prefer SR to LDP for next-hop resolution to other New York routers. Routers outside the region remain without SR and traffic continues to be forwarded without disruption.

Class of Service Preservation

Now that our island is functional, let's take a brief look at how SR supports networks that offer strict SLAs around uptime and class of service (CoS). A widely deployed technique is to disable penultimate hop popping (PHP) in favor of ultimate hop pop-like (UHP) behavior. Since SR does not directly advertise node SIDs as labels, it uses flags to request UHP behavior from upstream routers. Let's enable UHP on pe1.nyc and review the change to its neighbors' FIB.

Enable explicit-null UHP behavior on egress router pe1.nyc

```
user@pe1.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 1;
      }
      explicit-null;
    }
    level 1 labeled-preference 8;
  }
}
```

Control plane: verify the change in pe1.nyc's IS-IS LSP

```
user@pe1.nyc> show isis database pe1.nyc extensive
IS-IS level 1 link-state database:

pe1.nyc.00-00 Sequence: 0x4d4, Checksum: 0x4f97, Lifetime: 1195 secs
  IPV4 Index: 1
...
  IP extended prefix: 128.49.106.1/32 metric 0 up
    14 bytes of subtlvs
    Administrative tag 1: 1111
    Node SID, Flags: 0x70(R:0,N:1,P:1,E:1,V:0,L:0), Algo: SPF(0), Value: 1
```

In contrast with earlier output, pe1.nyc now advertises its node SID with both the 'P' (PHP-Off, or disable PHP) and 'E' (enable explicit-null, or use reserved label 0) flags set.

This leads its neighbors, p1.nyc and p2.nyc, to use the continue (swap) instead of the next (pop) action for label 1001 (pe1.nyc's node index is 1, and the network uses a common SRGB of 1K as discussed earlier). There is no change to pe2.nyc's FIB. It is not directly connected to pe1.nyc. PHP and UHP behaviors are only relevant for the immediately upstream routers.

Forwarding plane: verify pe1.nyc neighbors' FIB

```
user@p1.nyc> show route label 1001

mpls.0: 35 destinations, 35 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1001                *[L-ISIS/8] 00:04:03, metric 10
                    > to 192.0.2.4 via ge-0/0/2.0, Swap 0
1001(S=0)            *[L-ISIS/8] 00:04:03, metric 10
                    > to 192.0.2.4 via ge-0/0/2.0, Pop

user@p2.nyc> show route label 1001

mpls.0: 40 destinations, 40 routes (40 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1001                *[L-ISIS/8] 00:04:05, metric 10
                    > to 192.0.2.6 via ge-0/0/2.0, Swap 0
1001(S=0)            *[L-ISIS/8] 00:04:05, metric 10
                    > to 192.0.2.6 via ge-0/0/2.0, Pop

user@pe2.nyc> show route label 1001

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

1001                *[L-ISIS/8] 09:14:56, metric 20
                    to 192.0.2.9 via ge-0/0/1.0, Swap 1001
                    > to 192.0.2.11 via ge-0/0/2.0, Swap 1001
```

Both p1 and p2 are now swapping label 1001 with label 0, which is the well-known IPv4 explicit null label. This label ensures the MPLS traffic class (previously also known as experimental bits) is consistently copied to the last hop router. At that last hop, label 0 is popped and an IPv4 lookup occurs. The existence of label 0 has enabled CoS transparency and preservation.

Before proceeding, note that UHP is also enabled on pe2.nyc.

Traffic Protection and Restoration

Packet networks have long been able to deliver protection within 50ms of failure, crossing the bar set by SONET/SDH networks. IP-based fast reroute (FRR) is available for LDP as loop-free alternate (LFA) and remote LFA (rLFA). In turn, RSVP-TE has rich mechanisms in place to protect against loss of link, node, or fate-sharing resources identified as *shared-risk link groups* (SRLG).

Neither approach is without its caveats. LFA is topology-dependent in its ability to provide protection; rLFA improves on this at the expense of added targeted LDP session state and complexity. RSVP-TE has the most thorough feature set, which depends on additional signaling and state maintenance of bypass and detour LSPs.

SR's ability to source route, explicitly steering, and lack of transit control plane state lends itself well to establishing costless local protection paths. As long as there is physical redundancy in a topology, Topology-Independent Loop-free Alternate (TI-LFA) will find alternate routes. It can reroute around protected links, nodes, or locally-defined fate-sharing groups. Another improvement over erstwhile protection technologies is that TI-LFA protection paths are also ECMP-capable.

TI-LFA protection is local to the point of local repair (PLR). The backup paths are pre-computed but not pre-signaled. There is no singular merge point (MP) as each release point can be per-prefix optimized. The pre-computed paths prune or heavily cost out the protected resource from the topology in order to calculate post-IGP-convergence path(s). A failure of the protected resource results in a single update of the data plane, combining both local and global repair into one step.

Let's enable link protection on p1.nyc and observe the resulting change on its SR neighbors.

Enable TI-LFA link protection for all level 1 links on p1.nyc

```
user@p1.nyc> show configuration protocols isis
...
backup-spf-options {
    use-post-convergence-lfa;
}
...
interface all {
```

```

point-to-point;
level 1 {
    post-convergence-lfa;
}
}

```

Control plane: verify p1.nyc is announcing its adjacency SIDs as protection-eligible

```

user@p1.nyc> show isis database extensive p1.nyc level 1
IS-IS level 1 link-state database:

```

```

p1.nyc.00-00 Sequence: 0x510, Checksum: 0x192f, Lifetime: 1078 secs
  IPv4 Index: 3
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
  IS neighbor: pe2.nyc.00 Metric: 10
    Two-way fragment: pe2.nyc.00-00, Two-way first fragment: pe2.nyc.00-00
    P2P IPv4 Adj-SID: 47, Weight: 0, Flags: -BVL--
  IS neighbor: pe1.nyc.00 Metric: 10
    Two-way fragment: pe1.nyc.00-00, Two-way first fragment: pe1.nyc.00-00
    P2P IPv4 Adj-SID: 48, Weight: 0, Flags: -BVL--
  IS neighbor: p2.nyc.00 Metric: 10
    Two-way fragment: p2.nyc.00-00, Two-way first fragment: p2.nyc.00-00
    P2P IPv4 Adj-SID: 35, Weight: 0, Flags: -BVL-

```

The ‘B’ flag signals that p1.nyc considers these adjacency SIDs candidates for protection. Let’s look at one of these in detail – SID 47, which represents the link to pe2.nyc.

Forwarding plane: verify p1.nyc has installed backup paths for protection-eligible SIDs

```

user@p1.nyc> show route label 47 extensive

mpls.0: 35 destinations, 35 routes (35 active, 0 holddown, 0 hidden)
...
      Next hop: 192.0.2.8 via ge-0/0/3.0 weight 0x1, selected
      Label operation: Pop
...
      Next hop: 192.0.2.13 via ge-0/0/1.0 weight 0xf000
      Label operation: Swap 1000

user@p1.nyc> show isis adjacency
Interface      System      L State      Hold (secs) SNPA
ge-0/0/1.0     p2.nyc      3 Up          23
...
ge-0/0/3.0     pe2.nyc      1 Up          20

```

There are two unequal next hops, the latter newly added as a backup. Note that p1.nyc’s computed backup to reach pe2.nyc, avoiding the protected ge-0/0/3 link, is via p2.nyc. The primary action for incoming label 47 would be to pop and send directly to pe2.nyc. The backup action is to swap it for pe2.nyc’s node SID (label 1000) and send to p2.nyc, as shown in Figure 1.3. At that router, the label is swapped for 0, as a result of pe2.nyc advertising its reachability via explicit-null.

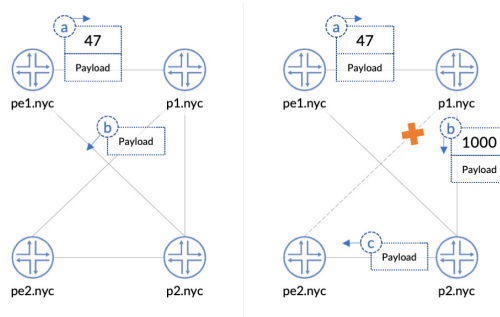


Figure 1.3 TI-LFA Avoiding Failed Link Between p1.nyc and pe2.nyc

Forwarding plane: verify p2.nyc acts as a transit router on this stateless protection path

```
user@p2.nyc> show route label 1000
```

```
mpls.0: 37 destinations, 37 routes (37 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
1000          *[L-ISIS/8] 12:18:02, metric 10
               > to 192.0.2.10 via ge-0/0/3.0, Swap 0
1000(S=0)     *[L-ISIS/8] 00:03:43, metric 10
               > to 192.0.2.10 via ge-0/0/3.0, Pop
```

This concludes the first leg: SR is enabled and in active use, albeit in a contained part of the network. There is reachability, and the ability to preserve CoS markings, as well as offer stateless traffic protection and restoration.

The next step is to broach SR into the rest of the network.

Chapter 2

Interoperability

As you know from Chapter 1, New York has transitioned to using SR for all intra-region traffic. Extra-region traffic still relies on LDP, which continues to run on all New York routers. This chapter reduces our dependence on LDP, while just as before, service continuity remains paramount.

As confidence is gained in the SR implementation, LDP can be eliminated entirely. This chapter focuses on eventually arriving at a state where LDP is no longer advertising unicast IPv4 or IPv6 FECs.

SR and LDP Interworking

LDP continues to distribute labels for all routers, even if the SR prefix SIDs are preferred in New York. Giving SR a higher preference only comes into effect if there is a competing route learned via LDP – when the the routers outside New York only have labels advertised via LDP, traffic to those destinations remains non-segment routed. If the plan is to turn LDP off, you must ensure you maintain a contiguous labeled path throughout, as LDP does for now (see Figure 2.1). This matters in both the SR-to-LDP as well as LDP-to-SR directions.

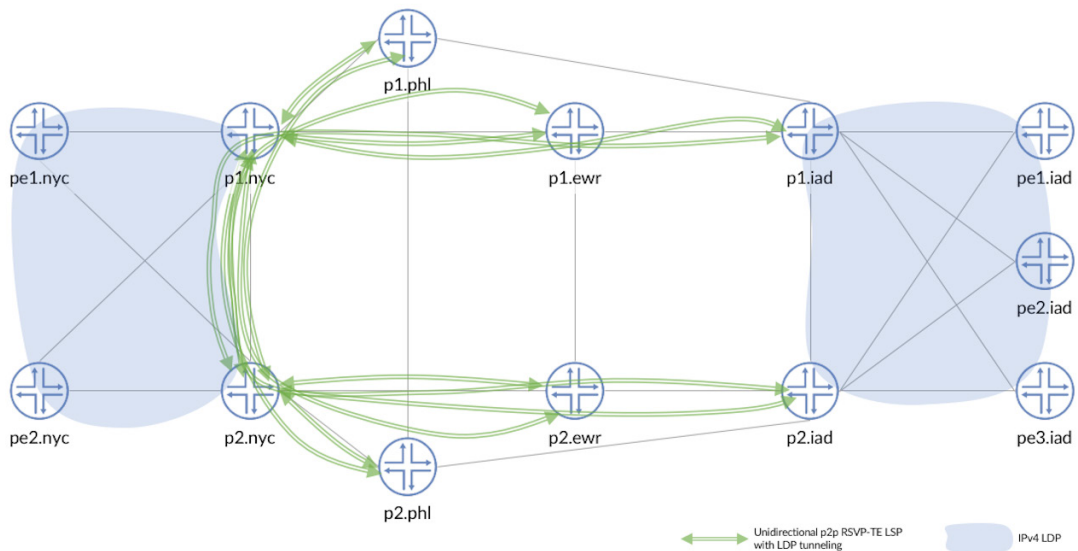


Figure 2.1 LDP at the Edge (IS-IS L1 areas) and Tunneled Over RSVP-TE Mesh in Core (IS-IS L2 subdomain)

SR-to-LDP

SR requires all routers to be reachable by a SID. Most of the network is lacking on that front. There are no prefix SIDs for remote routers as they are not yet running SR at all. Eliminate LDP in New York, and there will be no contiguous LSP outside the region. Clearly, then, you need a mechanism that indirectly associates prefix SIDs with non-SR-capable routers.

The *SR mapping server* (SRMS) is just such a control plane function. It advertises prefix SIDs on behalf of the non-SR-capable. As you would expect by now, this results in additional IS-IS TLVs that SR speakers will consume, and non-speakers ignore. SR routers will install prefix SIDs learned from mapping servers, much as they install native prefix SIDs, into inet.3.

Like all proxies, the SRMS must be highly available and its disseminated information must remain consistent – high availability is achieved by configuring more than one router as an SRMS. Mapping consistency is the aspirational result of following modern operational best practices, including templated configuration generation, version-controlled configuration repositories, and the ability to swiftly roll back unintended changes.

NOTE Complex tie-breaking rules exist if mapping inconsistencies occur. Each SRMS mapping carries a preference. Mappings from the SRMS with the highest preference are selected. If multiple SRMS have equal preference but advertise conflicting information, SR label collision rules come into effect.

The SRMS exists purely to advertise prefix to SID mappings. The prefixes themselves may or may not be reachable via the SRMS; in fact, the prefixes may be completely unreachable. Of course, advertising SID mappings for fictitious prefixes would be considered a misconfiguration, so standard change review process controls must be in effect to ensure incorrect mappings don't come to life.

The analog to a server is a client. For a mapping server to hold any sway, there must exist a mapping client (SRMC). Different from most protocols, SR mapping clients and servers don't form a distinct control plane relationship with each other. Instead, clients simply use the advertised mappings. An SRMS can simultaneously be both a server, as well as a client.

Routers bordering both domains are responsible for stitching the segments routed with LDP. The border router speaks both SR and LDP; on receiving a mapping, it creates a label swap entry for an incoming node SID to be switched with the LDP-learned entry for the same FEC in the mpls.0 table.

In this book's network, both p1.nyc and p2.nyc will act as border routers. While it may be more intuitive to first configure an SRMS, that would disrupt connectivity. Because the New York routers prefer SR over LDP, learning SR mapping entries for the Washington FECs would immediately supplant the existing LDP-learned labels in the PE's inet.3 table. Without explicitly enabling SR-to-LDP stitching behavior, when pe1.nyc and pe2.nyc attempt to use SR for long-haul traffic, traffic would be discarded at p1.nyc and p2.nyc whose mpls.0 table would lack a corresponding action for the inbound SR label.

Enabling SR to LDP Stitching

Let's configure both p1.nyc and p2.nyc as border routers, acting as SR/LDP translators:

```
user@p1.nyc> show configuration
protocols {
  isis {
    source-packet-routing ldp-stitching;
  }
}
```

There should be no change to forwarding behavior yet.

Connectivity verification: cross-country traffic still uses LDP labels

```

user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 4.390 ms 2.219 ms 2.002 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 43.247 ms 115.612 ms 94.366 ms
   MPLS Label=28 CoS=0 TTL=1 S=1
 3 p2.ewr-ge-0-0-2.0 (192.0.2.17) 11.741 ms p1.phl-ge-0-0-2.0 (192.0.2.21) 44.496 ms 75.404 ms
   MPLS Label=18 CoS=0 TTL=1 S=0
   MPLS Label=17 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-6.0 (192.0.2.28) 7.315 ms p1.iad-ge-0-0-5.0 (192.0.2.30) 6.626 ms 6.255 ms
   MPLS Label=17 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 129.904 ms 190.350 ms 100.185 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 9.483 ms 10.870 ms 8.107 ms

```

Here, both p1.nyc and p2.nyc are eagerly awaiting mapping entries from an SRMS. To start, p1.nyc will be configured as our first SRMS. It will advertise mappings for the PEs in Washington. Remember that any SR-capable router can act as an SRMS – its topological placement is irrelevant. A not strictly true comparison can be made with BGP route reflectors (in large topologies, judicious route reflector placement or the use of optimal route reflection becomes necessary):

```

user@p1.nyc> show configuration
routing-options {
  source-packet-routing {
    mapping-server-entry srms.p1 {
      prefix-segment-range 128.49.106/24 {
        start-prefix 128.49.106.10/32;
        start-index 10;
        size 4;
      }
    }
  }
}
...
protocols {
  isis {
    source-packet-routing mapping-server srms.p1;
  }
}

```

The policy is created in a protocol-independent section of the configuration. That policy is then referenced under the protocol-specific hierarchy to allow p1.nyc to act as an SRMS using IS-IS.

NOTE Remember that SR is agnostic to the choice of IGPs – it would work just as well, had our network been OSPF-based.

Let's tease this configuration apart. The policy name is a local matter. Within that, we are advertising a mapping entry that encompasses a range of four IPv4 prefixes, starting with 128.49.106.10 (pe1.iad's router-id).

This is efficiently encoded as a single label-binding TLV in p1.nyc's IS-IS LSP. In the next verification, the Range corresponds to the size keyword. The IPv4 prefix is our starting prefix. What we configured as the start-index is represented as Value.

Control plane: verify p1.nyc is acting as an SRMS

```

user@pe1.nyc> show isis database p1.nyc extensive level 1
...
Label binding: 128.49.106.10/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 4
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 10
...

```

The resulting entries in inet.3 tables of SRMC should be those in Table 2.1.

Table 2.1 p1.nyc SRMS Mapping Entries

Prefix	Label (SRGB + index)	Reachable?
128.49.106.10	1010 (1000 + 10)	Yes
128.49.106.11	1011 (1000 + 11)	Yes
128.49.106.12	1012 (1000 + 12)	No
128.49.106.13	1013 (1000 + 13)	Yes
128.49.106.10	1010 (1000 + 10)	Yes
128.49.106.11	1011 (1000 + 11)	Yes

Forwarding plane: verify pe1.nyc prefers SR to LDP

```

user@pe2.nyc> show route table inet.3 match-prefix "128.49.106.1?/32"

```

```

inet.3: 8 destinations, 14 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

128.49.106.10/32  *[L-ISIS/8] 00:11:32, metric 40
                  > to 192.0.2.9 via ge-0/0/1.0, Push 1010
                  > to 192.0.2.11 via ge-0/0/2.0, Push 1010
                  [LDP/9] 00:12:08, metric 40, tag 1111
                  > to 192.0.2.9 via ge-0/0/1.0, Push 27
                  > to 192.0.2.11 via ge-0/0/2.0, Push 27
128.49.106.11/32  *[L-ISIS/8] 00:11:32, metric 40
                  > to 192.0.2.9 via ge-0/0/1.0, Push 1011
                  > to 192.0.2.11 via ge-0/0/2.0, Push 1011
                  [LDP/9] 00:12:08, metric 40, tag 1111
                  > to 192.0.2.9 via ge-0/0/1.0, Push 28
                  > to 192.0.2.11 via ge-0/0/2.0, Push 28
128.49.106.13/32  *[L-ISIS/8] 00:11:32, metric 40
                  > to 192.0.2.9 via ge-0/0/1.0, Push 1013
                  > to 192.0.2.11 via ge-0/0/2.0, Push 1013
                  [LDP/9] 00:12:08, metric 40, tag 1111
                  > to 192.0.2.9 via ge-0/0/1.0, Push 29
                  > to 192.0.2.11 via ge-0/0/2.0, Push 29

```

As you can see, the LDP routes have been ignored in favor of the newly-learned SR prefix SIDs from p1.nyc acting as the SRMS. Where pe2.nyc would previously push the LDP label learned from p1.nyc or p2.nyc, it now uses the familiar SR labels starting at 1000.

The real crux however is seeing what p1.nyc and p2.nyc do when they receive these inbound labels. Since they are border routers, they should swap the mapping server derived label for the LDP-learned label. More specifically, the border routers should swap the inbound SR label for the outgoing label stack associated with the same FEC. Without that crucial step, you would have non-contiguous label switched paths.

Let's first make note of p1.nyc's outbound label stack towards p1.iad and how it's learned via LDP.

Forwarding plane: verify p1.nyc's outgoing label stack for p1.iad's FEC

```
user@p1.nyc> show route 128.49.106.11/32 table inet.3 detail
inet.3: 8 destinations, 13 routes (8 active, 0 holddown, 0 hidden)
128.49.106.11/32 (1 entry, 1 announced)
  State: <FlashAll>
    *LDP      Preference: 9
              Next hop type: Router, Next hop index: 0
              Address: 0xcc3ad10
              Next-hop reference count: 4
              Next hop: 192.0.2.21 via ge-0/0/5.0 weight 0x1, selected
              Label-switched-path p1.iad:0
              Label operation: Push 17, Push 18(top)
              Label TTL action: prop-ttl, prop-ttl(top)
              Load balance label: Label 17: None; Label 18: None;
              Label element ptr: 0xccdbaa0
              Label parent element ptr: 0xccdb2c0
              Label element references: 2
              Label element child references: 1
              Label element lsp id: 0
              Session Id: 0x0
              Next hop: 192.0.2.15 via ge-0/0/4.0 weight 0x8001
              Label-switched-path p1.iad:0
              Label operation: Push 17, Push 20(top)
              Label TTL action: prop-ttl, prop-ttl(top)
              Load balance label: Label 17: None; Label 20: None;
              ...
```

Since MPLS LSPs can be recursive, this is clearly a case of tunneling. Label 17 is learned from p1.nyc's LDP neighbor, who is its chosen next hop towards this FEC. That neighbor is p1.iad, reachable via the RSVP LSP over which an LDP session is tunneled.

Control plane: verify how p1.nyc created the label stack to reach p1.iads

```
user@p1.nyc> show rsvp session name p1.iad:0 ingress
Ingress RSVP: 4 sessions
To          From          State   Rt Style Labelin Labelout LSPname
128.49.106.9 128.49.106.3 Up       0  1 SE      -       18 p1.iad:0
Total 1 displayed, Up 1, Down 0

user@p1.nyc> show ldp database session 128.49.106.9
Input label database, 128.49.106.3:0--128.49.106.9:0
```

Labels received: 9

Label	Prefix
20	128.49.106.0/32
21	128.49.106.1/32
22	128.49.106.2/32
23	128.49.106.3/32
16	128.49.106.8/32
3	128.49.106.9/32
19	128.49.106.10/32
17	128.49.106.11/32
18	128.49.106.13/32

...

Based on this output, we should expect p1.nyc to swap incoming label 1011 with 17 (SR to LDP stitching), before pushing 18 on top (tunneled over RSVP LSP being exported as a forwarding adjacency).

Forwarding plane: confirm p1.nyc swaps the SR label for the LDP/RSVP combination

```
user@p1.nyc> show route label 1011 detail
```

```
mpls.0: 41 destinations, 41 routes (41 active, 0 holddown, 0 hidden)
1011 (1 entry, 1 announced)
  *L-ISIS Preference: 14
    Level: 2
    Next hop type: Router, Next hop index: 0
    Address: 0xcc3b290
    Next-hop reference count: 1
    Next hop: 192.0.2.21 via ge-0/0/5.0 weight 0x1, selected
    Label-switched-path p1.iad:0
    Label operation: Swap 17, Push 18(top)
    Label TTL action: prop-ttl, prop-ttl(top)
    Load balance label: Label 17: None; Label 18: None;
    Label element ptr: 0xccdd6c0
    Label parent element ptr: 0x0
    Label element references: 2
    Label element child references: 0
    Label element lsp id: 0
    Session Id: 0x0
    Next hop: 192.0.2.15 via ge-0/0/4.0 weight 0x8001
    Label-switched-path p1.iad:0
    Label operation: Swap 17, Push 20(top)
    Label TTL action: prop-ttl, prop-ttl(top)
    Load balance label: Label 17: None; Label 20: None;
    ...
```

Voilà! The label operation for both the FEC in inet.3 and the corresponding label in mpls.0 are identical. When pe1.nyc pushes label 1011 on traffic destined for pe1.iad, p1.nyc (and p2.nyc) swap that for the LDP label (that an RSVP label is pushed next is incidental).

The key is how well SR coexists with the classic label distribution protocols, both the namesake LDP, as well as the powerful RSVP.

Let's run another long-distance traceroute. It should confirm that pe1.nyc and pe2.nyc now use the SRMS advertised mapping for pe1.iad. Of course, pe1.iad, as with other non-New York routers, remains SR-oblivious.

Connectivity verification: pe2.nyc now pushes SR label 1011, instead of the earlier LDP-learned label

```
user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 3.418 ms 2.418 ms 3.593 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 12.553 ms 6.212 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 6.776 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 3 p1.phl-ge-0-0-2.0 (192.0.2.21) 9.093 ms 7.572 ms p2.ewr-ge-0-0-2.0 (192.0.2.17) 8.630 ms
   MPLS Label=16 CoS=0 TTL=1 S=0
   MPLS Label=16 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-6.0 (192.0.2.28) 13.157 ms 7.112 ms 12.586 ms
   MPLS Label=16 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 11.872 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 8.175 ms 5.957 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 8.596 ms 16.851 ms 7.024 ms
```

We would be remiss if we didn't validate the lack of expected change to intra-New York traffic.

Connectivity verification: traffic within New York continues to use SR

```
user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 9.106 ms 1.801 ms 1.862 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 6.196 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 94.158 ms 91.205 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-2.0 (192.0.2.6) 42.152 ms pe1.nyc-ge-0-0-1.0 (192.0.2.4) 70.788 ms 67.294 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 129.316 ms 135.363 ms 146.201 ms
```

Excellent. There is no change from the earlier traceroutes.

Before proceeding, let's address the lack of SRMS redundancy by configuring p2.nyc as our second mapping server. You can simply configure it identically to p1.nyc:

```
user@p2.nyc> show configuration
routing-options {
  source-packet-routing {
    mapping-server-entry srms.p2 {
      prefix-segment-range 128.49.106/24 {
        start-prefix 128.49.106.10/32;
        start-index 10;
        size 4;
      }
    }
  }
}
protocols {
  isis {
    source-packet-routing mapping-server srms.p2;
  }
}
```

Control plane: p2.nyc also originates a label-binding SID with the same mapping as p1.nyc

```
user@p2.nyc> show isis database p2.nyc extensive level 1
...
Label binding: 128.49.106.10/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 4
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 10
```

There is no change to forwarding behavior. If p1.nyc becomes unavailable, the mappings from p2.nyc continue to appear as surrogate node SIDs.

For the sake of understanding a different approach to configuring SRMS entries, let's remove this configuration from p2.nyc, and instead, add individual prefix entries and then investigate how they are differently encoded in the IS-IS LSP, yet lead to the same forwarding behavior:

```
user@p2.nyc> show configuration
routing-options {
  source-packet-routing {
    mapping-server-entry srms.p2 {
      prefix-segment 128.49.106.10/32 index 10;
      prefix-segment 128.49.106.11/32 index 11;
      prefix-segment 128.49.106.13/32 index 13;
    }
  }
}
protocols {
  isis {
    source-packet-routing mapping-server srms.p2;
  }
}
```

We have created mappings for individual prefixes, instead of a range. Unlike the range that covered a prefix that isn't currently part of our network (128.49.106.12), this form allows non-contiguous mappings. As you would expect, it requires separate TLVs in p2.nyc's IS-IS PDU.

Control plane: p2.nyc now has multiple label binding TLVs

```
user@pe1.nyc> show isis database p2.nyc extensive
...
TLVs:
Area address: 49.0001 (3)
LSP Buffer Size: 1492
Speaks: IP
Speaks: IPV6
IP router id: 128.49.106.2
IP address: 128.49.106.2
Label binding: 128.49.106.10/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 10
Label binding: 128.49.106.11/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 11
Label binding: 128.49.106.13/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 13
...
```

Where before we saw a single TLV provide mappings for all Washington's PE routers (as well as an extraneous entry), we now find three TLVs, one for each PE. If the prefixes are not contiguous, this form will be needed. Keep in mind that both approaches have their own considerations:

- Prefix ranges are the most compact way to distribute this information. Their scope may be too broad, covering prefixes that should or do not need SRMS services.
- Multiple, label-binding TLVs are more precise but lead to inflated IGP PDUs, whose flooding and fragmentation may make distribution less efficient.

Of course, both approaches can be simultaneously used – a range when prefixes are conveniently addressed, and separate entries when inconvenient.

Conflict Resolution

Let's explore this hybrid approach with a novel idea – advertise mappings for SR-capable nodes, from p2.nyc. The p2.nyc SRMS will continue to advertise individual entries for Washington routers, and add a range that covers the New York routers. The intent is simply to understand whether this causes any confusion or harm:

```
user@p2.nyc> show configuration routing-options
source-packet-routing {
  mapping-server-entry srms.p2 {
    prefix-segment 128.49.106.10/32 index 10;
    prefix-segment 128.49.106.11/32 index 11;
    prefix-segment 128.49.106.13/32 index 13;
    prefix-segment-range 128.49.106/24 {
      start-prefix 128.49.106.0/32;
      start-index 0;
      size 4;
    }
  }
}
```

You can see our existing entries augmented by a range starting with pe2.nyc's router-id (128.49.106.0). Since pe2.nyc is already advertising its node index, and the index matches what the SRMS purports to originate, there is no change to the forwarding behavior.

Control plane: p2.nyc is advertising ranges for both SR-capable and incapable routers

```
user@pe2.nyc> show isis database p2.nyc extensive
...
Label binding: 128.49.106.10/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 10
Label binding: 128.49.106.11/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
```

```

Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 11
Label binding: 128.49.106.13/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 13
Label binding: 128.49.106.0/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 4
Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 0
...

```

The exercise served to educate us that both individual and prefix ranges can be used together. While not damaging, this configuration does not represent an operational best practice. Muddying our intent with gratuitous advertisements is, at best, confusing, and at worst, a landmine for future software behavior or standards specification changes.

Remove the unneeded prefix-segment-range.

LDP-to-SR

Have we now eliminated the need for LDP on all New York PE routers? They have reachability to each other via SR node SIDs; they have additionally learned synthetic node SIDs for the Washington PE routers. Is LDP still necessary? After one last step, we will be through with it on the PEs.

Remember that the Washington routers learn about New York FECs via LDP. It is tunneled over the RSVP LSPs originating at p1.nyc and p2.nyc, and terminating at p1.iad and p2.iad. LDP can't be turned off on these routers without breaking the contiguous LSP.

We can turn LDP off on pe1.nyc and pe2.nyc. Let's first think about how we can continue to advertise labels for them via LDP to the Washington routers. Since the LDP session between the P and PE routers would be removed, the P routers need to be configured to originate an LDP label on the behalf of the PEs.

This is the same idea as SRMS, turned the other way around, through the proverbial looking glass. Instead of advertising prefix SIDs on behalf of the SR-incapable, we need to advertise proxy FEC mappings for the LDP-averse.

Before making changes, let's verify the LDP labels being advertised to reach pe2.nyc by p2.nyc.

Control plane: p2.nyc is advertising label 56 to its LDP neighbors

```

user@p2.nyc> show ldp database | match "put|128.49.106.0"
Input label database, 128.49.106.2:0--128.49.106.0:0
 3      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.0:0
56      128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.1:0
32      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.1:0
56      128.49.106.0/32

```

```

Input label database, 128.49.106.2:0--128.49.106.3:0
 34      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.3:0
 56      128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.8:0
 51      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.8:0
 56      128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.9:0
 47      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.9:0
 56      128.49.106.0/32

```

Forwarding plane: p2.nyc pops label 56 as its performing PHP to pe2.nyc

```
user@p2.nyc> show route label 56
```

```

mpls.0: 40 destinations, 40 routes (40 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

56          *[LDP/9] 00:04:53, metric 1, tag 1111
              > to 192.0.2.10 via ge-0/0/3.0, Pop
56(S=0)     *[LDP/9] 00:04:53, metric 1, tag 1111
              > to 192.0.2.10 via ge-0/0/3.0, Pop

```

Connectivity verification: traffic from Washington to pe2.nyc uses label 56

```

user@ce1.iad> traceroute wait 1 198.51.100.2 routing-instance svc-inet
traceroute to 198.51.100.2 (198.51.100.2), 30 hops max, 40 byte packets
 1 pe1.iad-ge-0-0-11.0 (198.51.100.55) 38.571 ms 2.340 ms 1.981 ms
 2 p1.iad-ge-0-0-2.0 (192.0.2.37) 7.628 ms p2.iad-ge-0-0-2.0 (192.0.2.39) 9.786 ms p1.iad-
ge-0-0-2.0 (192.0.2.37) 6.620 ms
   MPLS Label=47 CoS=0 TTL=1 S=1
 3 p1.ewr-ge-0-0-3.0 (192.0.2.27) 99.309 ms 93.491 ms 94.582 ms
   MPLS Label=31 CoS=0 TTL=1 S=0
   MPLS Label=34 CoS=0 TTL=1 S=1
 4 p2.nyc-ge-0-0-4.0 (192.0.2.16) 6.603 ms 7.013 ms 7.230 ms
   MPLS Label=56 CoS=0 TTL=1 S=1
 5 pe2.nyc-ge-0-0-2.0 (192.0.2.10) 10.157 ms pe2.nyc-ge-0-0-1.0 (192.0.2.8) 10.194 ms 9.562 ms
 6 ce1.nyc-ge-0-0-2.1 (198.51.100.2) 14.194 ms 42.463 ms 7.111 ms

```

Enabling LDP to SR Stitching

Now let's enable the LDP to SR stitching functionality at p2.nyc and observe the change in the advertised labels:

```

user@p2.nyc> show configuration
protocols {
  ldp {
    sr-mapping-client;
  }
}

```

Control plane: p2.nyc allocates new LDP label 59, withdrawing label 56

```

user@p2.nyc> show ldp database | match "put|128.49.106.0"
Input label database, 128.49.106.2:0--128.49.106.0:0
  3      128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.0:0
  59     128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.1:0
  32     128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.1:0
  59     128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.3:0
  34     128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.3:0
  59     128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.8:0
  54     128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.8:0
  59     128.49.106.0/32
Input label database, 128.49.106.2:0--128.49.106.9:0
  47     128.49.106.0/32
Output label database, 128.49.106.2:0--128.49.106.9:0
  59     128.49.106.0/32

```

Control plane: LDP-to-SR stitching functionality is enabled

```

user@p2.nyc> show ldp overview
Instance: master
  Reference count: 6
  Router ID: 128.49.106.2
  LDP inet: enabled
...
  LDP SR Mapping Client: enabled
...

```

Forwarding plane: label 59 is swapped with 0 as pe2.nyc requests SR UHP treatment

```

user@p2.nyc> show route label 59

mpls.0: 40 destinations, 40 routes (40 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

59          *[LDP/9] 00:01:20, metric 1, tag 1111
              > to 192.0.2.10 via ge-0/0/3.0, Swap 0
59(S=0)     *[LDP/9] 00:01:20, metric 1, tag 1111
              > to 192.0.2.10 via ge-0/0/3.0, Pop

```

Connectivity verification: label 59 is now in use by p2.nyc's LDP neighbors to reach pe2.nyc

```

user@ce1.iad> traceroute wait 1 198.51.100.2 routing-instance svc-inet
traceroute to 198.51.100.2 (198.51.100.2), 30 hops max, 40 byte packets
 1  pe1.iad-ge-0-0-11.0 (198.51.100.55) 55.677 ms 96.798 ms 83.824 ms
 2  p1.iad-ge-0-0-2.0 (192.0.2.37) 91.130 ms 81.951 ms p2.iad-ge-0-0-2.0 (192.0.2.39) 42.568 ms
    MPLS Label=54 CoS=0 TTL=1 S=1
 3  p1.ewr-ge-0-0-3.0 (192.0.2.27) 132.320 ms p2.ewr-ge-0-0-3.0 (192.0.2.29) 7.874 ms 6.459 ms

```

```

MPLS Label=16 CoS=0 TTL=1 S=0
MPLS Label=59 CoS=0 TTL=1 S=1
4 p2.nyc-ge-0-0-4.0 (192.0.2.16) 7.673 ms p1.nyc-ge-0-0-4.0 (192.0.2.14) 123.033 ms p2.nyc-
ge-0-0-4.0 (192.0.2.16) 7.327 ms
MPLS Label=59 CoS=0 TTL=1 S=1
5 pe2.nyc-ge-0-0-1.0 (192.0.2.8) 521.158 ms pe2.nyc-ge-0-0-2.0 (192.0.2.10) 64.296 ms pe2.
nyc-ge-0-0-1.0 (192.0.2.8) 183.317 ms
6 ce1.nyc-ge-0-0-2.1 (198.51.100.2) 9.445 ms 10.869 ms 11.214 ms

```

You can enable the LDP SR mapping client – what’s been referred to as *LDP-to-SR stitching* – as well as on p1.nyc to ensure there are multiple paths into New York. Then you are free to turn LDP off on both pe1.nyc and pe2.nyc. They will no longer advertise an LDP label for themselves, but p1.nyc and p2.nyc will continue to do so as shown in Figure 2.2.

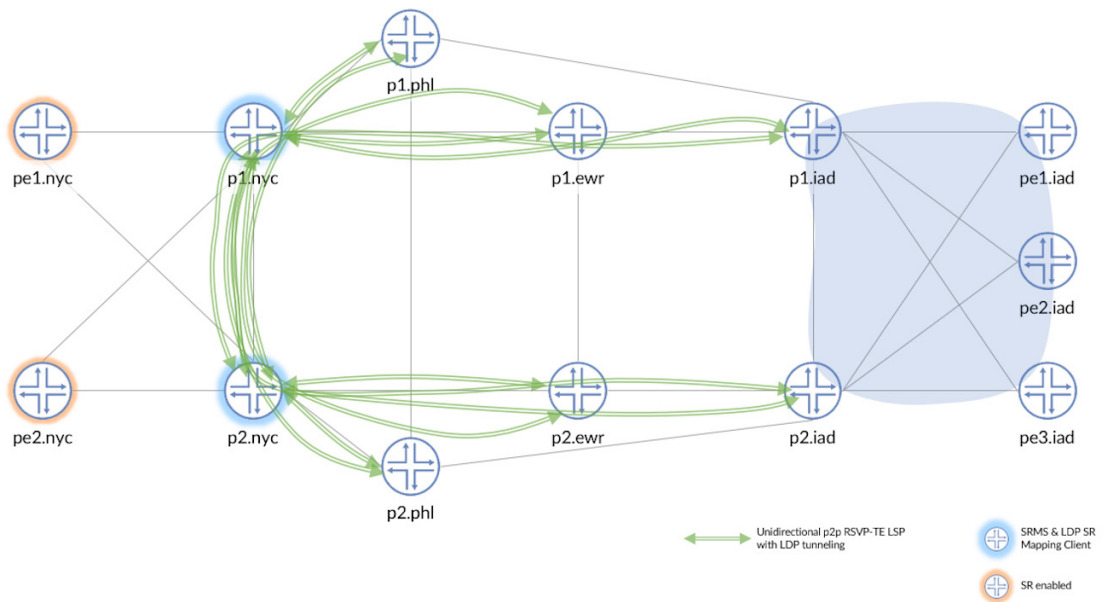


Figure 2.2

LDP Eliminated on New York PE Routers

You can persist for as long as needed in this state. This can be a legitimate mode of operation if the routers outside New York are not going to be SR-capable for the foreseeable future. *Inadequate hardware or software capabilities in one part of the network should not prevent an SR deployment.* Well planned and executed though the change needs to be, it does not need to be a big bang event that requires all platforms simultaneously converting to segment routing.

SR and RSVP-TE

It was briefly touched upon that a full mesh of LDP-tunneling RSVP-TE LSPs exists between the New York and the Washington P routers. At the end of the previous section, LDP was eliminated from pe1.nyc and pe2.nyc. However, LDP remained in use at p1.nyc, p2.nyc, and their Washington counterparts.

Interworking: SR over RSVP-TE

Let's plow forward by mirroring the changes we have wrought in New York:

1. Enable SR on all Washington routers.
2. Prefer SR to LDP at the Washington PEs.

Once all the ingress routers will have switched over to SR, we can then finally eradicate LDP. That will include the SRMS and LDP SR mapping client configured on p1.nyc and p2.nyc, as illustrated in Figure 2.3.

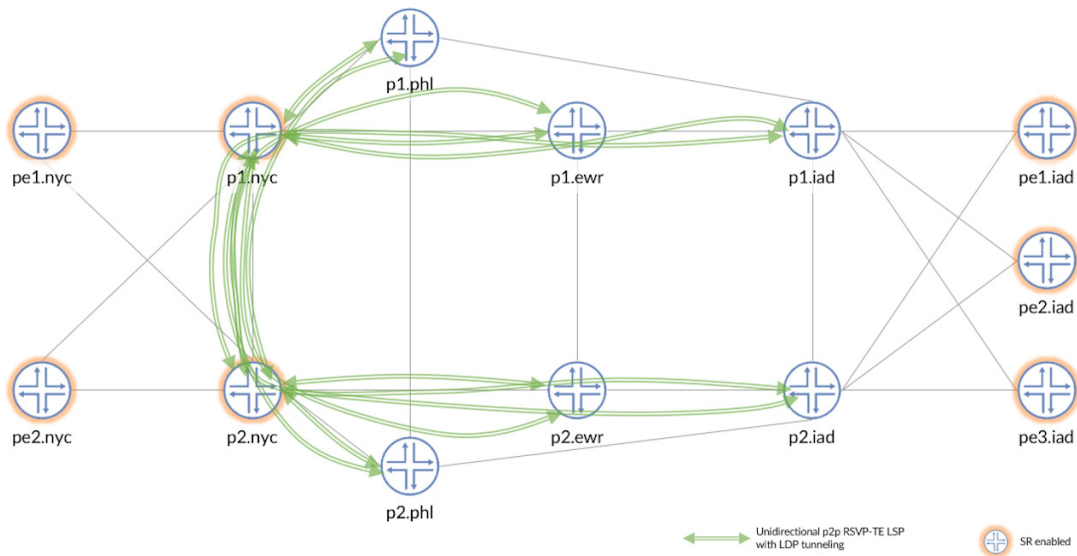


Figure 2.3

LDP Eliminated On All Routers

The importance of prefix SID planning cannot be overstated. The New York routers learn these via the pair of SRMS. Native node SID configuration should retain identical values. In fact, SRGB sizing, offset, and SID planning are fundamental to the success of any segment routing deployment. Do not rush into a live deployment without careful consideration of use cases, as well as the ability of extant equipment to use identical, and consistently-sized, SRGB.

You also need to be careful about the order in which you enable SR functionality in Washington. If it is first enabled on the P routers, connectivity will break. This is because p1.nyc and p2.nyc will determine their Washington neighbors are suddenly SR-capable. They will stop swapping the SRMS label, which they advertise for pe1.nyc and pe2.nyc to push, with the LDP label learned across the RSVP-TE LSP. The SR-to-LDP stitching functionality will be deemed unnecessary.

Instead, SR is first enabled on all the Washington PEs:

```
user@pe1.iad> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 11;
      }
    }
  }
}
...
user@pe2.iad> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 10;
      }
    }
  }
}
...
user@pe3.iad> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 13;
      }
    }
  }
}
...
```

Let's check to ensure there is no disruption.

Connectivity verification: traffic forwarding within New York is unchanged

```
user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 4.094 ms 2.933 ms 2.475 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 6.052 ms 5.527 ms p1.nyc-ge-0-0-3.0 (192.0.2.9) 5.449 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-2.0 (192.0.2.6) 9.073 ms 5.860 ms 4.729 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 6.433 ms 7.107 ms 5.952 ms
```

Connectivity verification: traffic forwarding cross-country is unchanged

```
user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 181.515 ms 211.310 ms 123.021 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 7.206 ms p1.nyc-ge-0-0-3.0 (192.0.2.9) 6.855 ms p2.nyc-
ge-0-0-3.0 (192.0.2.11) 6.535 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 3 p2.phl-ge-0-0-2.0 (192.0.2.23) 6.944 ms p1.ewr-ge-0-0-2.0 (192.0.2.15) 7.656 ms 10.006 ms
   MPLS Label=45 CoS=0 TTL=1 S=0
```

```

MPLS Label=17 CoS=0 TTL=1 S=1
4 p2.iad-ge-0-0-5.0 (192.0.2.32) 7.512 ms p1.iad-ge-0-0-6.0 (192.0.2.26) 12.778 ms p2.iad-
ge-0-0-5.0 (192.0.2.32) 6.871 ms
MPLS Label=22 CoS=0 TTL=1 S=1
5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 6.567 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 6.934 ms pe1.
iad-ge-0-0-1.0 (192.0.2.36) 6.115 ms
6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 8.500 ms 7.762 ms 7.320 ms

```

No change is good (in this case). Washington is a separate Level 1 area, and the L1/L2 P routers are not yet SR-enabled. SR-to-LDP stitching continues unabated. The moment SR is enabled on p1.iad and p2.iad, the need for stitching will become moot.

This may seem gratuitous, but it's worth thinking about how the need for stitching is determined. If a neighbor doesn't advertise SR capability, p1.nyc and p2.nyc will merrily swap an SR label for LDP. In our network, they will additionally push an RSVP-TE LSP label atop.

If p1.iad (or p2.iad) is then configured with a node SID, both p1.nyc and p2.nyc will see its refreshed IS-IS Level 2 LSP. Correctly, they will no longer consider it for SR-to-LDP stitching as it is evident that p1.iad (or p2.iad) will be able to forward SR-native labels.

This is exactly how to proceed, with one wrinkle. We'll cost out the Washington P router which is not being enabled for SR first. This will ensure that all cross-region traffic will be segment routed; not only that, it will be carried over RSVP-TE LSPs, demonstrating SR over RSVP-TE.

Arbitrarily, let's enable SR on p2.iad and overload p1.iad:

```

user@p1.iad> show configuration
protocols {
  isis {
    overload;
  }
}

```

This forces all Washington-bound traffic towards p2.iad.

Connectivity verification: traffic forwarding to Washington is no longer multi-pathed

```

user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 24.153 ms 42.585 ms 85.331 ms
2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 7.771 ms 7.338 ms 7.904 ms
MPLS Label=1011 CoS=0 TTL=1 S=1
3 p2.phl-ge-0-0-2.0 (192.0.2.23) 6.619 ms 8.563 ms 6.279 ms
MPLS Label=41 CoS=0 TTL=1 S=0
MPLS Label=22 CoS=0 TTL=1 S=1
4 p2.iad-ge-0-0-5.0 (192.0.2.32) 45.022 ms 66.523 ms 67.692 ms
MPLS Label=22 CoS=0 TTL=1 S=1
5 pe1.iad-ge-0-0-2.0 (192.0.2.38) 9.016 ms 7.956 ms 5.477 ms
6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 63.975 ms 67.474 ms 65.506 ms

```

Connectivity verification: traffic forwarding within New York remains unaffected

```
user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 2.712 ms 2.005 ms 2.196 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 5.053 ms p1.nyc-ge-0-0-3.0 (192.0.2.9) 5.036 ms p2.nyc-
ge-0-0-3.0 (192.0.2.11) 6.494 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-1.0 (192.0.2.4) 4.992 ms pe1.nyc-ge-0-0-2.0 (192.0.2.6) 5.301 ms pe1.nyc-
ge-0-0-1.0 (192.0.2.4) 5.028 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 6.404 ms 10.624 ms 5.638 ms
```

Forwarding plane: both New York P routers swap the inbound SR label for LDP-o-RSVP

```
user@p1.nyc> show route label 1011 extensive
mpls.0: 44 destinations, 44 routes (44 active, 0 holddown, 0 hidden)
...
1011 /52 -> {list:Swap 22, Push 48(top), Swap 22, Push 43, Push 91(top)}
```

```
user@p2.nyc> show route label 1011 extensive
mpls.0: 48 destinations, 48 routes (48 active, 0 holddown, 0 hidden)
...
1011 /52 -> {list:Swap 22, Push 41(top), Swap 22, Push 39(top)}
```

This output shows both routers swap 1011 for 22, the LDP label advertised by p2.iad over the LDP session tunneled over RSVP-TE. The RSVP-TE LSP label is then pushed. There are two unequal cost paths, the latter representing the RSVP-TE bypass.

SR is then enabled on p2.iad:

```
user@p2.iad> show configuration
protocols {
  isis {
    source-packet-routing {
      srgb start-label 1000 index-range 128;
      node-segment {
        ipv4-index 8;
      }
    }
  }
}
```

The NY P routers continue to avoid p1.iad. And rather than swapping 1011 for an LDP label, they simply swap for the next SR instruction. That happens to be idempotent, a *continue* action, resulting in 1011 being swapped for 1011.

Forwarding plane: both New York P routers now swap the inbound SR label for another SR label

```
user@p1.nyc> show route label 1011 extensive
mpls.0: 43 destinations, 43 routes (43 active, 0 holddown, 0 hidden)
...
1011 /52 -> {list:Swap 1011, Push 48(top), Swap 1011, Push 43, Push 91(top)}
```

```

user@p2.nyc> show route label 1011 extensive
mpls.0: 46 destinations, 46 routes (46 active, 0 holddown, 0 hidden)
...
1011 /52 -> {list:Swap 1011, Push 41(top), Swap 1011, Push 39(top)}

```

Note that the RSVP-TE LSP pushed labels don't change. It is the bottom label that is no longer stitching between SR and LDP. Let's verify that connectivity remains, and that SR is now correctly carried across the RSVP-TE core.

Connectivity verification: no change within New York as expected

```

user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 3.562 ms 2.242 ms 2.280 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 5.827 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 5.176 ms p1.nyc-
ge-0-0-3.0 (192.0.2.9) 4.909 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-1.0 (192.0.2.4) 5.346 ms 4.597 ms 6.729 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 6.242 ms 7.008 ms 6.032 ms

```

Connectivity verification: outside New York, SR labels are carried all the way until pe1.iad, not just the L1/L2

```

user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 2.707 ms 2.235 ms 1.859 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 7.356 ms 6.876 ms 6.823 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 3 p2.ph1-ge-0-0-2.0 (192.0.2.23) 79.374 ms 99.533 ms 100.238 ms
   MPLS Label=41 CoS=0 TTL=1 S=0
   MPLS Label=1011 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-5.0 (192.0.2.32) 8.332 ms 5.620 ms 6.191 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-2.0 (192.0.2.38) 6.701 ms 6.149 ms 5.633 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 50.015 ms 80.652 ms 16.288 ms

```

This is geographically farther than we have gotten so far with SR. Until now, the complex interplay of stitching and LDP tunneling has allowed SR traffic to be carried across the RSVP-TE LSPs without additional explicit configuration.

As we have learned, making our intent explicit is both a noble goal and an operational best practice. Platform defaults age poorly, dating themselves. The less reliant an operator is on orthodox behavior – once desirable, later antiquated – the more robust the network. Explicating is both self-documenting and a way of enforcing consistency across platforms, software releases, and the people responsible for them.

Let's rectify this by configuring SR to be explicitly carried over RSVP-TE LSPs. This should cause no immediate change, but once LDP is removed, will ensure we maintain connectivity. On both P routers in New York and Washington, let's enable SR shortcuts:

```

user@p1.nyc> show configuration
protocols {
    isis {
        traffic-engineering {
            family inet-mpls {
                shortcuts;
            }
        }
    }
}

```

Connectivity verification: no change in either traffic flow

```

user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 200.634 ms 131.682 ms 348.936 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 44.074 ms 61.865 ms 89.987 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 3 p2.ewr-ge-0-0-2.0 (192.0.2.17) 69.335 ms 84.690 ms 85.139 ms
   MPLS Label=74 CoS=0 TTL=1 S=0
   MPLS Label=1011 CoS=0 TTL=1 S=1
 4 p2.iad-ge-0-0-6.0 (192.0.2.28) 7.209 ms 8.750 ms 7.218 ms
   MPLS Label=1011 CoS=0 TTL=1 S=1
 5 pe1.iad-ge-0-0-2.0 (192.0.2.38) 6.394 ms 6.114 ms 5.724 ms
 6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 7.612 ms 7.749 ms 7.764 ms

```

```

user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 2.493 ms 1.882 ms 2.250 ms
 2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 4.085 ms p1.nyc-ge-0-0-3.0 (192.0.2.9) 7.766 ms 5.468 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.iad-ge-0-0-2.0 (192.0.2.38) 6.799 ms 4.908 ms pe1.nyc-ge-0-0-1.0 (192.0.2.4) 4.566 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 6.375 ms 5.636 ms 6.442 ms

```

All that remains now is to remove LDP (including SRMS and stitching in both directions on the New York P routers, as well as LDP tunneling on the RSVP-TE LSPs), enable SR on p1.iad, and cost it back into operation.

These steps have been previously discussed, dissected, and detailed. They don't need to be belabored. When completed, it will have made good on the promise to have replaced LDP with SR. Throughout we have seen how well the segment routing architecture fits in with the MPLS label distribution protocols.

Connectivity verification: complete reachability intra- and inter-region using SR

```

user@ce1.nyc> traceroute wait 1 198.51.100.0 routing-instance svc-inet
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 2.517 ms 2.329 ms 2.028 ms
 2 p1.nyc-ge-0-0-3.0 (192.0.2.9) 5.231 ms p2.nyc-ge-0-0-3.0 (192.0.2.11) 39.534 ms 62.640 ms
   MPLS Label=1001 CoS=0 TTL=1 S=1
 3 pe1.nyc-ge-0-0-2.0 (192.0.2.6) 7.699 ms pe1.nyc-ge-0-0-1.0 (192.0.2.4) 6.293 ms pe1.nyc-ge-0-0-2.0 (192.0.2.6) 4.781 ms
 4 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 7.089 ms 6.082 ms 6.172 ms

```

```

user@ce1.nyc> traceroute wait 1 198.51.100.54 routing-instance svc-inet
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe2.nyc-ge-0-0-10.1 (198.51.100.3) 102.157 ms 209.904 ms 134.732 ms

```

```

2 p2.nyc-ge-0-0-3.0 (192.0.2.11) 8.376 ms 7.249 ms 7.538 ms
  MPLS Label=1011 CoS=0 TTL=1 S=1
3 p2.ewr-ge-0-0-2.0 (192.0.2.17) 9.670 ms 7.303 ms 7.385 ms
  MPLS Label=74 CoS=0 TTL=1 S=0
  MPLS Label=1011 CoS=0 TTL=1 S=1
4 p2.iad-ge-0-0-6.0 (192.0.2.28) 7.269 ms p1.iad-ge-0-0-6.0 (192.0.2.26) 8.827 ms p2.iad-
ge-0-0-6.0 (192.0.2.28) 11.046 ms
  MPLS Label=1011 CoS=0 TTL=1 S=1
5 pe1.iad-ge-0-0-1.0 (192.0.2.36) 7.451 ms 6.142 ms pe1.iad-ge-0-0-2.0 (192.0.2.38) 6.463 ms
6 ce1.iad-ge-0-0-4.0 (198.51.100.54) 65.857 ms 68.233 ms 63.493 ms

```

```

user@ce1.iad> traceroute routing-instance svc-inet wait 1 198.51.100.0
traceroute to 198.51.100.0 (198.51.100.0), 30 hops max, 40 byte packets
1 pe1.iad-ge-0-0-11.0 (198.51.100.55) 55.924 ms 64.637 ms 70.194 ms
2 p1.iad-ge-0-0-2.0 (192.0.2.37) 110.243 ms 61.692 ms 57.564 ms
  MPLS Label=1001 CoS=0 TTL=1 S=1
3 p1.phl-ge-0-0-3.0 (192.0.2.31) 60.380 ms p2.ewr-ge-0-0-3.0 (192.0.2.29) 35.849 ms 7.796 ms
  MPLS Label=75 CoS=0 TTL=1 S=0
  MPLS Label=1001 CoS=0 TTL=1 S=1
4 p1.nyc-ge-0-0-5.0 (192.0.2.20) 9.929 ms 6.796 ms 7.237 ms
  MPLS Label=1001 CoS=0 TTL=1 S=1
5 pe1.nyc-ge-0-0-2.0 (192.0.2.6) 6.479 ms pe1.nyc-ge-0-0-1.0 (192.0.2.4) 6.491 ms 12.843 ms
6 ce1.nyc-ge-0-0-1.1 (198.51.100.0) 8.924 ms 7.647 ms 9.697 ms

```

```

user@ce1.iad> traceroute routing-instance svc-inet wait 1 198.51.100.2
traceroute to 198.51.100.2 (198.51.100.2), 30 hops max, 40 byte packets
1 pe1.iad-ge-0-0-11.0 (198.51.100.55) 2.657 ms 2.126 ms 2.304 ms
2 p1.iad-ge-0-0-2.0 (192.0.2.37) 6.891 ms 6.503 ms p2.iad-ge-0-0-2.0 (192.0.2.39) 7.041 ms
  MPLS Label=1000 CoS=0 TTL=1 S=1
3 p2.ewr-ge-0-0-3.0 (192.0.2.29) 49.937 ms 60.684 ms p1.phl-ge-0-0-3.0 (192.0.2.31) 82.859 ms
  MPLS Label=73 CoS=0 TTL=1 S=0
  MPLS Label=1000 CoS=0 TTL=1 S=1
4 p2.nyc-ge-0-0-4.0 (192.0.2.16) 7.331 ms p1.nyc-ge-0-0-5.0 (192.0.2.20) 85.649 ms p2.nyc-
ge-0-0-4.0 (192.0.2.16) 7.287 ms
  MPLS Label=1000 CoS=0 TTL=1 S=1
5 pe2.nyc-ge-0-0-1.0 (192.0.2.8) 84.588 ms 43.379 ms 7.002 ms
6 ce1.nyc-ge-0-0-2.1 (198.51.100.2) 8.398 ms 7.586 ms 8.608 ms

```

With SR successfully transported over the traffic engineered core, let's make note of the fact that, once again, this could remain a perpetual mode of operation. The likelihood of multi-pathing is greater in-region, where SR has been deployed. Where multipathing is poorer – such as across regions – RSVP-TE is exercised. The two can coexist perennially.

For our purposes, though, nothing holds us back from enabling SR in the entire L2 subdomain. The configuration is similar to previous examples and does not need to be repeated. Since our objective to demonstrate interworking has been met, let's also decommission the RSVP-TE LSPs between p1 and p2 in New York and Washington so that all traffic within and between is entirely transported using SR.

Coexistence: SR-aware Bandwidth Reservation with RSVP-TE

RSVP-TE has a staggering feature set, accumulated over decades of real-world operational experience. One particularly well-known and widely-used feature is auto-bandwidth: resizing and potentially rerouting LSPs to match the offered traffic load. Auto-bandwidth can appear almost magical, with next to no manual input, save the initial configuration parameters, and an optimal path can be cleaved.

Grossly, auto-bandwidth relies on two values: the LSP reservation size, and the available bandwidth on a given interface in the network. The first is derived by the LSP ingress router using periodic rate measurement, smoothed over an interval. The latter is disseminated throughout the IGP area by all routers as bandwidth is consumed or released. The two form a feedback loop that drives the distributed auto-bandwidth calculation.

Implicit in this Swiss clockwork affair is the assumption that RSVP-TE is exclusively used to transport traffic. The available interface bandwidth is stored in the traffic engineering database (TED), whose utilization snapshot is populated only by RSVP-TE. If the interface also carries non-RSVP-TE traffic, one unappetizing stopgap is to cap RSVP-TE reservable bandwidth to a fraction of an interface's capacity. This will prevent RSVP-TE LSPs from competing with other users of that bandwidth, such as non-MPLS traffic, at the expense of possible link under-utilization.

NOTE RFC8426 (<https://datatracker.ietf.org/doc/rfc8426/>) details this dark bandwidth problem statement, as well as other approaches to tackle it. Partitioning bandwidth is no panacea. There is no guarantee non-RSVP-TE users will limit themselves naturally to their apportioned capacity.

In our network, SR-MPLS traffic would be a consumer of bandwidth, alongside RSVP-TE. In order for the two to coexist, at least until a complete migration occurs, SR traffic utilization must be reflected in the TED's maximum reservable interface bandwidth. Thus RSVP-TE LSPs are indirectly made aware that they may not have dedicated access to capacity, without needing to statically partition bandwidth.

While we have cost out the RSVP-TE LSPs encountered so far, there remain RSVP-TE LSPs that carry traffic between non-SR-capable devices outside Washington and New York. For brevity's sake, let's not focus too deeply on these routers or LSPs – indeed, they aren't even represented in the diagram. Instead, let's simply ensure the TEDs for all routers portray an accurate view of the network's available bandwidth.

While there are no more RSVP-TE LSPs originating and terminating between New York and Washington, others continue to persist between other regions. Some of the same interfaces used by RSVP-TE LSPs also switch SR traffic, now natively. So let's configure p1 and p2 in New York and Washington to measure SR traffic and deduct that from the remaining interface bandwidth (available to RSVP-TE):

```

routing-options {
  auto-bandwidth {
    template {
      SR-RSVP-COEXIST {
        adjust-interval 30;
        adjust-threshold 1;
        statistic-collection-interval 10;
      }
    }
  }
}
protocols {
  isis {
    source-packet-routing {
      traffic-statistics {
        statistics-granularity per-interface;
        auto-bandwidth SR-RSVP-COEXIST;
      }
    }
  }
}
...

```

The `collection-interval` represents how frequently statistics are culled. The `adjust interval` is the duration of the window across which counter samples are collected. The average of those samples – in our case that would be three samples – represents the current SR traffic load per-link. If this load exceeds the `adjust threshold`, which is the delta between the current and previously computed average, the IGP floods the maximum reservable bandwidth, which refreshes the TED.

The result may involve RSVP-TE LSPs being preempted, rerouted, as well as the IGP flooding the revised maximum reservable bandwidth for the TE link. With no SR traffic in flight, let's see how much bandwidth is available for reservation on the link between p1.nyc to p1.phl.

Control plane: 3Kbps in use, 97Kbps available to RSVP-TE

```

user@p1.nyc> show rsvp interface ge-0/0/5.0

```

Interface	State	Active resv	Subscr- ption	Static BW	Available BW	Reserved BW
ge-0/0/5.0	Up	9	100%	100kbps	97kbps	3kbps

Now let's crank up traffic that is segment routed and observe the reduction in reported available bandwidth.

Control plane: ~67Kbps used by SR

```

user@p1.nyc> show auto-bandwidth traffic detail ge-0/0/5
Name: ge-0/0/5.0
Collection Interval: 10, Adjust Interval: 30, Adjust Threshold: 1%
Adjust Subscription: 100%

```

```

Pkt Recv: 154.098k, Byte Recv: 13.5604M, Query Count: 238, Average: 66.926kbps
Last Base Bytes: 83.658k, Last Report Time: Tue Jan 15 17:50:44
Last Query Time: Tue Jan 15 17:50:44
Last Resp Time: Tue Jan 15 17:50:44
Byte Bucket(Chronological order, first entry is latest):
  96.536k 109.032k 45.408k
Packet Bucket(Chronological order, first entry is latest):
  1.097k 1.239k 516

```

Control plane: 31Kbps available to RSVP-TE, reduced by ~67Kbps

```

user@p1.nyc> show rsvp interface ge-0/0/5.0

```

Interface	State	Active resv	Subscr-ption	Static BW	Available BW	Reserved BW
ge-0/0/5.0	Up	9	100%	100kbps	31kbps	3kbps

Control plane: LSDB reflects updated bandwidth

```

user@p1.nyc> show isis database p1.nyc extensive level 2
IS-IS level 2 link-state database:
...
IS extended neighbor: p1.phl.00, Metric: default 10 SubTLV len: 81
IP address: 192.0.2.20
Neighbor's IP address: 192.0.2.21
Local interface index: 337, Remote interface index: 334
Current reservable bandwidth:
  Priority 0 : 31kbps
  Priority 1 : 31kbps
  Priority 2 : 31kbps
  Priority 3 : 31kbps
  Priority 4 : 31kbps
  Priority 5 : 31kbps
  Priority 6 : 31kbps
  Priority 7 : 31kbps
Maximum reservable bandwidth: 34kbps
Maximum bandwidth: 100kbps
...

```

This reflection works in reverse, too. Let's staunch the SR traffic and observe how the available bandwidth once again increases.

Control plane: SR utilization drops to ~40Kbps

```

user@p1.nyc> show auto-bandwidth traffic detail ge-0/0/5
Name: ge-0/0/5.0
Collection Interval: 10, Adjust Interval: 30, Adjust Threshold: 1%
Adjust Subscription: 100%
Pkt Recv: 235.006k, Byte Recv: 20.6803M, Query Count: 317, Average: 39.846kbps
Last Base Bytes: 54.325k, Last Report Time: Tue Jan 15 18:03:44
Last Query Time: Tue Jan 15 18:03:54
Last Resp Time: Tue Jan 15 18:03:54
Byte Bucket(Chronological order, first entry is latest):
  0 63.184k 86.24k
Packet Bucket(Chronological order, first entry is latest):
  0 718 980

```

Control plane: RSVP-TE now indicates 54Kbps available

```
user@p1.nyc> show rsvp interface ge-0/0/5.0
```

Interface	State	Active resv	Subscr- ption	Static BW	Available BW	Reserved BW
ge-0/0/5.0	Up	9	100%	100kbps	54kbps	3kbps

Control plane: The LSDB values are flooded to neighbors & match RSVP-TE's reported values

```
user@p1.nyc> show isis database p1.nyc extensive level 2
IS-IS level 2 link-state database:
...
IS extended neighbor: p1.phl.00, Metric: default 10 SubTLV len: 81
IP address: 192.0.2.20
Neighbor's IP address: 192.0.2.21
Local interface index: 337, Remote interface index: 334
Current reservable bandwidth:
  Priority 0 : 54kbps
  Priority 1 : 54kbps
  Priority 2 : 54kbps
  Priority 3 : 54kbps
  Priority 4 : 54kbps
  Priority 5 : 54kbps
  Priority 6 : 54kbps
  Priority 7 : 54kbps
Maximum reservable bandwidth: 57kbps
Maximum bandwidth: 100kbps
...
```

This graceful coexistence of SR and RSVP-TE anticipates operator needs. For those with demanding deployments, RSVP-TE is unlikely to be quickly displaced. Ensuring the available bandwidth reflects more than one consumer allows accurate reservations.

SR and IPv6

Whatever your opinion on IPv6, you're probably right. If you fail to see a strong driver, one likely doesn't exist for the business you happen to be in; if you worship at the altar of *Happy Eyeballs*, it's no less than an imperative and technological differentiator.

NOTE *Happy Eyeballs* is a set of algorithms defined by the IETF. It aims for a superior user experience when using dual-stack hosts. IPv6 is preferred, where available. Check <https://tools.ietf.org/html/rfc8305> for details.

IPv6 support for existing MPLS label distribution protocols varies: LDPv6 extends the base protocol; RSVP, up until this date, doesn't offer a generally available implementation; BGP remains address family agnostic, but of course it isn't an IGP.

The good news is that segment routing treats IPv6 reachability as a first-class citizen. SR won't convince you to deploy IPv6 anew, but then, it also won't become a barrier to entry. Node, adjacency, and anycast SIDs (of which we've used the first two so far), come in both IPv4 and IPv6 flavors.

CAUTION Segment routing support for IPv6 is most commonly understood to mean associating SIDs with IPv6 prefixes. The data plane remains MPLS-switched. In contrast, SRv6 is a radical reimagining of native IPv6 forwarding. It does not make use of MPLS at all. SRv6 may be a topic for a separate book.

Anyway, the configuration is nearly identical so let's dive right in and create an IPv6 MPLS-encapsulated underlay. The syntax is shown for pe1.nyc. The SID allocation is detailed in Table 2.2 to avoid repeating the mostly identical configuration, and Figure 2.4 illustrates the setup.

```
protocols {
  isis {
    source-packet-routing {
      node-segment {
        ipv6-index 61;
```

Table 2.2 IPv6 Node SID Allocation

Router	IPv6 lo0 address	IPv6 node SID (index + SRGB)
pe1.nyc	2001:db8::128:49:106:1	1061 (61 + 1000)
pe2.nyc	2001:db8::128:49:106:0	1060 (60 + 1000)
p1.nyc	2001:db8::128:49:106:3	1063 (63 + 1000)
p2.nyc	2001:db8::128:49:106:2	1062 (62 + 1000)
pe1.iad	2001:db8::128:49:106:11	1071 (71 + 1000)
pe2.iad	2001:db8::128:49:106:10	1070 (70 + 1000)
pe3.iad	2001:db8::128:49:106:13	1073 (73 + 1000)
p1.iad	2001:db8::128:49:106:9	1069 (69 + 1000)
p2.iad	2001:db8::128:49:106:8	1068 (68 + 1000)

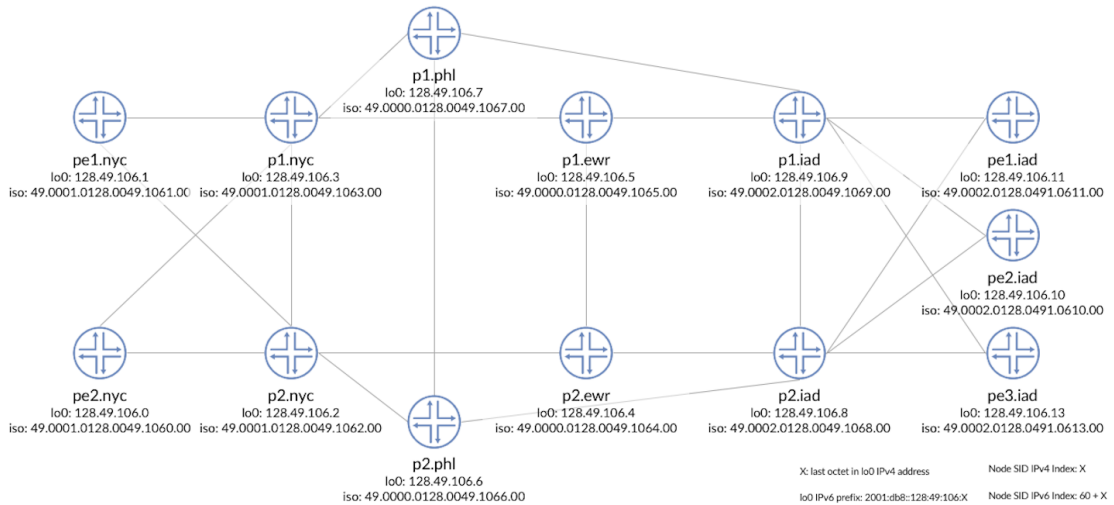


Figure 2.4 IPv6, ISO Addressing and Additional Node SID Numbering

As soon as this configuration becomes effective, each router additionally starts to advertise an IPv6 node index (alongside the IPv4 node index), as well as IPv6 adjacency SIDs.

Control plane: Additional IPv6 node index and adjacency SIDs

```
user@pe1.nyc> show isis database pe1.nyc extensive
IS-IS level 1 link-state database:

pe1.nyc.00-00 Sequence: 0x2a7, Checksum: 0x6dc8, Lifetime: 803 secs
  IPv4 Index: 1, IPv6 Index: 61
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 128, Label-Range: [ 1000, 1127 ]
    IS neighbor: p2.nyc.00 Metric: 10
      Two-way fragment: p2.nyc.00-00, Two-way first fragment: p2.nyc.00-00
      P2P IPv4 Adj-SID: 17, Weight: 0, Flags: --VL--
      P2P IPv6 Adj-SID: 36, Weight: 0, Flags: F-VL--
    IS neighbor: p1.nyc.00 Metric: 10
      Two-way fragment: p1.nyc.00-00, Two-way first fragment: p1.nyc.00-00
      P2P IPv4 Adj-SID: 25, Weight: 0, Flags: --VL--
      P2P IPv6 Adj-SID: 35, Weight: 0, Flags: F-VL--
  IP prefix: 128.49.106.1/32 Metric: 0 Internal Up
  V6 prefix: 2001:db8::128:49:106:1/128 Metric: 0 Internal Up
```

The 'F' flag in the adjacency SID indicates an IPv6-capable adjacency. Unsurprisingly, you'll see new entries in the recently populated inet6.3 routing table.

Control plane: FECs in inet6.3 use native addresses, not mapped IPv4 addresses

```
user@pe1.nyc> show route table inet6.3
```

```
inet6.3: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
2001:db8::128:49:106:0/128
    *[L-ISIS/8] 00:06:40, metric 20
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0, Push 1060
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0, Push 1060
2001:db8::128:49:106:2/128
    *[L-ISIS/8] 00:06:40, metric 10
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0
2001:db8::128:49:106:3/128
    *[L-ISIS/8] 00:06:40, metric 10
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0
2001:db8::128:49:106:8/128
    *[L-ISIS/8] 00:06:40, metric 30
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0, Push 1068
2001:db8::128:49:106:9/128
    *[L-ISIS/8] 00:06:40, metric 30
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0, Push 1069
2001:db8::128:49:106:10/128
    *[L-ISIS/8] 00:06:40, metric 40
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0, Push 1070
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0, Push 1070
2001:db8::128:49:106:11/128
    *[L-ISIS/8] 00:06:40, metric 40
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0, Push 1071
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0, Push 1071
2001:db8::128:49:106:13/128
    *[L-ISIS/8] 00:06:40, metric 40
    > to fe80::5668:a3ff:fe1e:4af5 via ge-0/0/1.0, Push 1073
    > to fe80::5668:a3ff:fe1e:4ab6 via ge-0/0/2.0, Push 1073
```

And let's verify connectivity to these new service routes, both intra-region as well as cross-region. You can see next that IPv6 service prefixes are being carried by IPv6 transport. 6PE's mapped IPv4 addresses and use of IPv6 explicit null can be laid to rest.

Connectivity verification: Intra-region using the new IPv6 node SIDs

```
user@ce1.nyc> traceroute 2001:db8::198:51:100:2
```

```
traceroute6 to 2001:db8::198:51:100:2 (2001:db8::198:51:100:2) from 2001:db8::198:51:100:0, 64 hops max, 12 byte packets
```

```
1 2001:db8::198:51:100:1 (2001:db8::198:51:100:1) 33.108 ms 11.671 ms 3.032 ms
2 p1.nyc-lo0.0 (2001:db8::128:49:106:3) 60.297 ms 73.248 ms p2.nyc-lo0.0 (2001:db8::128:49:106:2) 6.085 ms
```

```
MPLS Label=1060 CoS=0 TTL=1 S=1
```

```
3 pe2.nyc-lo0.0 (2001:db8::128:49:106:0) 8.127 ms 5.868 ms 6.411 ms
4 2001:db8::198:51:100:2 (2001:db8::198:51:100:2) 7.223 ms 7.043 ms 6.865 ms
```

Connectivity verification: inter-region using the new IPv6 node SIDs

```

user@ce1.iad> traceroute 2001:db8::198:51:100:0
traceroute6 to 2001:db8::198:51:100:0 (2001:db8::198:51:100:0) from 2001:db8::198:51:100:54, 64 hops
max, 12 byte packets
 1 2001:db8::198:51:100:55 (2001:db8::198:51:100:55) 3.297 ms 2.640 ms 2.752 ms
 2 p2.iad-lo0.0 (2001:db8::128:49:106:8) 8.982 ms p1.iad-lo0.0 (2001:db8::128:49:106:9) 8.719 ms
8.548 ms
    MPLS Label=1061 CoS=0 TTL=1 S=1
 3 p1.ewr-lo0.0 (2001:db8::128:49:106:5) 65.592 ms p2.phl-lo0.0 (2001:db8::128:49:106:6) 8.300 ms
7.831 ms
    MPLS Label=1061 CoS=0 TTL=1 S=1
 4 p1.nyc-lo0.0 (2001:db8::128:49:106:3) 8.081 ms p2.nyc-lo0.0 (2001:db8::128:49:106:2) 8.512 ms
8.452 ms
    MPLS Label=1061 CoS=0 TTL=1 S=1
 5 pe1.nyc-lo0.0 (2001:db8::128:49:106:1) 8.251 ms 8.593 ms 8.803 ms
 6 2001:db8::198:51:100:0 (2001:db8::198:51:100:0) 9.266 ms 8.870 ms 8.603 ms

```

SR and Multicast

A common myth is that multicast needs special treatment in a segment routed network. Like most myths, this stems from a lack of understanding.

Multicast forwarding is orthogonal to unicast. Existing approaches – IPv4 multicast, IPv6 multicast, MPLS multicast – remain usable even as SR is enabled. When it comes to MPLS multicast, it may indeed mean that LDP and RSVP-TE – two protocols that predate SR – are left intact for multicast forwarding.

While mLDP offers both point-to-multipoint (P2MP) and multipoint-to-multipoint (MP2MP) replication trees; RSVP-TE offers point-to-point (P2P) ingress replication, as well as P2MP network replication. Both support delivery of non-VPN and multicast VPN traffic.

SR-native approaches such as Spray and Tree-SID aspire to offer ingress and network replication, respectively. BIER is technically not segment routing but follows a similar ideal of eliminating state from transit nodes. Until these technologies mature, operators can rely on existing multicast mechanisms.

Chapter 3

Observability

Inertia can be overpowering. Supplanting a technology requires competing with not only the technology itself, but with its ecosystem. The ecosystem comprises people—network engineers, designers, architects, and operators who have become specialized using a particular toolkit. It includes ordering, billing, service assurance, orchestration, and visualization applications. These can slow the transition and even outright prevent it.

Luckily segment routing is enjoying a great deal of popularity. This has resulted in human attention being invested in developing, learning, and deploying it. Applications are catching up, offering familiar user experiences to flatten the learning curve; the best promise novel use cases that are uniquely achievable with SR.

This chapter explores tools that visualize, and provide service assurance for, a segment routed network.

Visualization

NorthStar is an MPLS underlay and selective overlay controller. Its genesis is rooted in the rise of centralization of traffic policy and control. Rather than aiming to be an illusory single pane of glass, it stays grounded by implementing well-defined traffic engineering functionality.

MORE? *Day One: NorthStar Controller Up and Running* is the best starter resource to learn more about this application: <https://www.juniper.net/us/en/training/jnbooks/day-one/networking-technologies-series/northstar-controller/>.

This focus of this chapter will not be administering NorthStar, but enabling an existing deployment to add value to an SR network.

In turn, we will also stay grounded. First, we'll visualize the network built so far. A new BGP address family – `link-state` – will be enabled between the area border routers (ABR) and NorthStar's Junos VM. This exports the topology for each area – nodes, links, link attributes, and prefixes. Both Junos and NorthStar have been enhanced to support export of SR-specific capabilities, such as the SRGB range, node, and adjacency SIDs.

Apply the configuration below to the ABRs in both New York and Washington – `p1.nyc`, `p2.nyc`, `p1.iad`, and `p2.iad`, substituting router-specific values as appropriate. This provides redundancy by configuring all the ABRs in a region to export area-specific topology; it also ensures coverage of the entire multi-level network:

```
policy-options {
  policy-statement NLRI-T0-BGP-LS {
    term 1 {
      from family traffic-engineering;
      then accept;
    }
  }
  policy-statement TED-T0-NLRI {
    term LEVEL-1 {
      from {
        protocol isis;
        level 1;
      }
      then accept;
    }
    term LEVEL-2 {
      from {
        protocol isis;
        level 2;
      }
      then accept;
    }
  }
}
protocols {
  bgp {
    group NORTHSTAR {
      type internal;
      local-address 192.168.88.3;
      family traffic-engineering {
        unicast;
      }
      export NLRI-T0-BGP-LS;
      neighbor 192.168.88.99;
    }
  }
  isis {
    traffic-engineering igp-topology;
  }
  mpls {
```

```

traffic-engineering {
  database {
    import {
      igp-topology {
        bgp-link-state;
      }
    }
    policy TED-T0-NLRI;
  }
}
...

```

Let's take a peek at what p1.nyc is exporting from both its areas. We'll review one example from each of the nodes, links, and prefixes and how that information has been enriched with SR attributes.

Control plane: p1.nyc exporting pe2.nyc as a node, as well as its SR attributes

```

user@p1.nyc> show route advertising-protocol bgp 192.168.88.99 te-node-iso 0128.0049.1060.00
lsdist.0: 82 destinations, 82 routes (82 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
  NODE { AS:4200000000 ISO:0128.0049.1060.00 ISIS-L1:0 }/1216
*
      Self
      100      I
      IPv4 Router-ids:
      128.49.106.0
      Area border router: No
      External router: No
      Attached: No
      Overload: Yes
      Hostname: pe2.nyc
      Area membership:
      49 00 01
      SPRING-Capabilities:
      - SRGB block [Start: 1000, Range: 128, Flags: 0xc0]
      SPRING-Algorithms:
      - Algo: 0

```

Control Plane: p1.nyc exporting the link between itself and pe1.nyc, and its local adjacency SID

```

user@p1.nyc> show route advertising-protocol bgp 192.168.88.99 te-link-local-ip 192.0.2.5
lsdist.0: 82 destinations, 82 routes (82 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
  LINK { Local { AS:4200000000 ISO:0128.0049.1063.00 }.
  { IPv4:192.0.2.5 } Remote { AS:4200000000 ISO:0128.0049.1061.00 }.
  { IPv4:192.0.2.4 } ISIS-L1:0 }/1216
*
      Self
      100      I
      Metric: 10
      TE Metric: 10
      P2P IPV4 Adj-SID - Label: 16, Flags: 0x70, Weight: 0

```

Control plane: p1.nyc exporting pe2.iad's prefix and associated SID index

```

user@p1.nyc> show route advertising-protocol bgp 192.168.88.99 te-ipv4-prefix-ip 128.49.106.10
lsdist.0: 82 destinations, 82 routes (82 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
  PREFIX { Node { AS:4200000000 ISO:0128.0049.1062.00 } { IPv4:128.49.106.10/32 } ISIS-L1:0 }/1216
*
      Self
      100      I
      Prefix SID: 10, Flags: 0xe0, Algo: 0

```

The sum total of the BGP-LS exchange allows NorthStar to construct a graph of the network as shown in Figures 3.1 and 3.2.

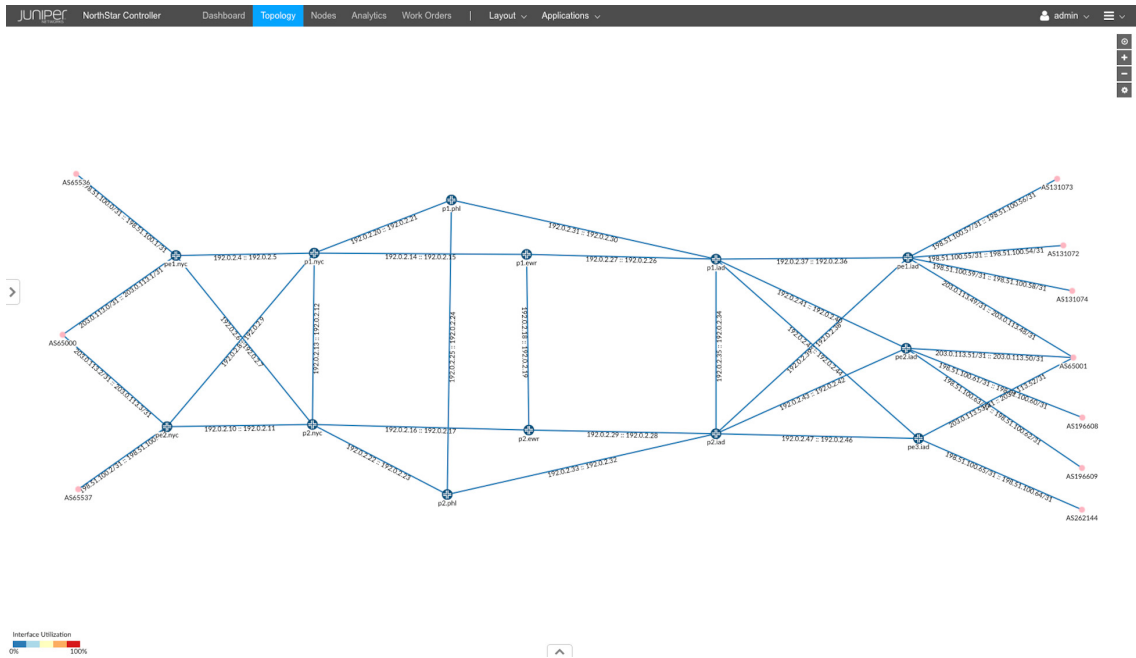


Figure 3.1 Topology View With ASNs Discovered After Device Configuration Parsing

Type	Name	Created	Frequency	Repeats	Starts	Ends	Last Executed	Status
ExecuteScript	CollectionCleanup	2019-02-07 20:02:21...	Daily	1	2019-02-08 04:00:00...	Never	2019-02-09 04:00:00...	Scheduled
ExecuteScript	ESRokupTask	2019-02-07 20:02:21...	Hourly	1	2019-02-07 20:15:00...	Never	2019-02-08 20:15:00...	Scheduled
Device Collection	getConfig	2019-02-08 10:24:57	Immediately	N/A	2019-02-08 10:24:57...	N/A	2019-02-08 10:25:00...	Completed

Figure 3.2 NorthStar Device Collection

Besides unmarshalling BGP-LS ‘routes,’ NorthStar can collect, via NETCONF, and parse device configuration. You can do this by traversing its menu at the top right (Administration > Task Scheduler > Add > Device Collection). Be sure to specify a non-root user’s credentials.

Collecting device configuration is optional but improves the user experience. For instance, the ASNs to which our network is attached become visible in the UI. You can drill into Layer 3 VPN VRFs to discover their PEs, attachment circuits, and route targets. These aren’t particulars exported by BGP-LS. Configuration parsing presents the operator with multiple aspects of their network.

But this is a book about segment routing. NorthStar has several valuable SR-specific visualizations. Each link can display not just an IP address, but also its locally-allocated adjacency SID. You can select this by clicking the gear icon in the workspace (below the +/- zoom buttons in Figure 3.3). From there, choose Links > Show Label > IP, SID A::Z.

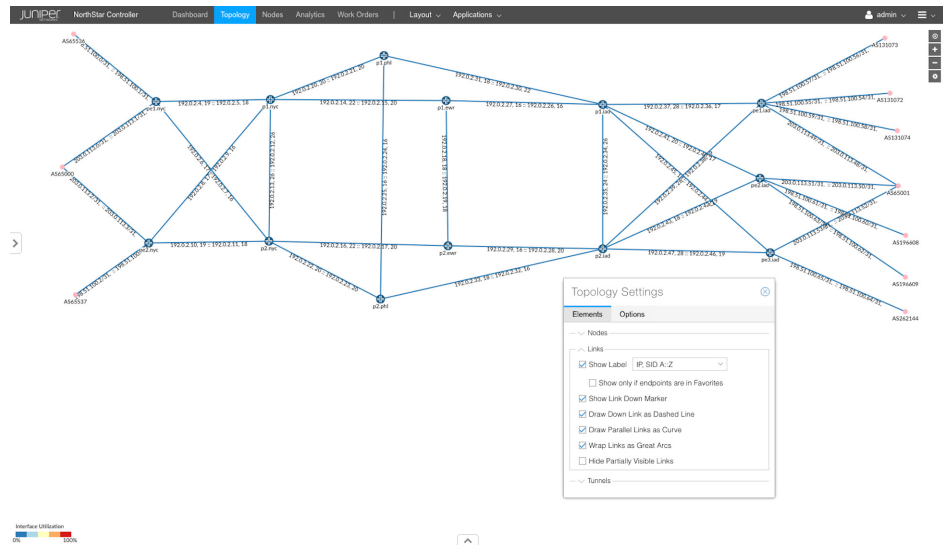


Figure 3.3

Northstar Topology View With Adjacency SIDs Selected

Another outstandingly useful feature is to right-click a router and select *Node SIDs from selected node*. The labels of the other router change to display the node SID alongside their hostname. This occurs by adding each node’s (IPv4-only at the moment) index to the advertised SRGB of the router’s neighbors.

As a best practice, the SRGB base and size have been kept consistent in our network. In networks where this isn’t or can’t be followed, this feature alone can be a lifesaver.

We'll pause our exploration of NorthStar as a visualization tool. We'll return to it in its day job as a traffic engineering controller. At that point, we'll build upon this view-only exercise by calculating and provisioning SR-TE policies (the proper name for SR-TE LSPs).

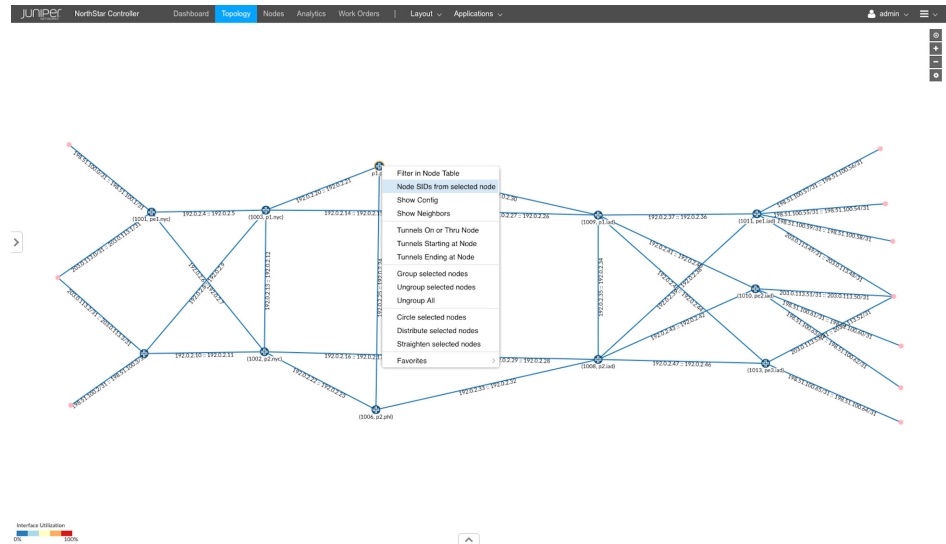


Figure 3.4 Northstar Shows Node SIDs Based On a Single Router's Point of View

Telemetry

Befitting its radical nature, Junos' SR implementation forgoes exporting any statistics via SNMP. Long known to be antiquated, SNMP is being superseded by vastly more capable protocols and frameworks that organize information.

On the protocol front, gRPC obsoletes SNMP. It inherits the benefits of HTTP/2, such as:

- **Binary:** Unlike text, serializing and un-serializing binary protocols is more efficient, compact, and less prone to errors over aspects such as white space handling. gRPC uses Protocol Buffers to encode and decode data. SNMP uses ASN.1.
- **Multiplexed:** This allows multiple outstanding requests over a single TCP connection, while still allowing for parallelism. In practice, SNMP is UDP-based and inherits its non-guaranteed nature.

- *Bidirectional*: Unlike its predecessor, HTTP/2 allows the server to push potentially useful data to the client, without being explicitly requested.

gRPC supports mutual certificate-based authentication; the rarely used SNMPv3 only supports shared key authentication. gRPC also offers both event-driven and periodic publish/subscribe approaches. This is similar to SNMP's polling of statistics and relying on unsolicited traps to convey state changes. What makes gRPC more powerful is that it allows a client to express interest in subsets of the state information tree; as that state changes, the server publishes the update to clients – this could include a router's ARP or ND table, LSP state, et al.

NOTE jtimon is an open-source gRPC client that's invaluable for testing. Not only does it support subscribing to multiple paths for both periodic and event-triggered changes, it also supports mutual authentication, convenient conversion of key/value pairs to JSON, and it exports to Influxdb and Prometheus. You can find it at <https://github.com/nileshsimaria/jtimon>.

Building upon this improved transport is OpenConfig, an initiative to model networks in a vendor-neutral manner. This is an attempt to improve upon SNMP's method of organizing information – the Management Information Base (MIB) – which resulted in sparse vendor-neutral support. Both configuration and operational data models have been defined by OpenConfig for the most widely used features – BGP, LLDP, and of course, SR. Others remain in the works.

NOTE Explore the Junos OpenConfig feature set at <https://juni.pr/2WRR1st>. Visit www.openconfig.net to keep up with, and possibly contribute to, the progress on data models. Remember that OpenConfig is not a single technology or protocol, but the name of a working group who shares the goal of more programmable infrastructure.

The robust transport protocol and intuitive data model representation makes modern telemetry easier for the operator.

For instance, this is how an SNMP poll for an interface's operational status would traditionally be done. First, the interface's ifIndex would have to be discovered. This is an arbitrary integer value assigned to an interface by the device. All interface-specific state and counters are keyed using that index. In long-standing environments the ifIndex would be cached, possibly reassigned, and consequently relearned on device reboot. Then the state and counters associated with that interface would have to be polled:

```
% snmpbulkwalk -v 2c -c public -Cr20 pe1.nyc.sr.q13.ak IF-MIB::ifDescr | grep ge-0/0/0
IF-MIB::ifDescr.527 = STRING: ge-0/0/0
```

```
% snmpget -v 2c -c public pe1.nyc.sr.q13.ak IF-MIB::ifOperStatus.527
IF-MIB::ifOperStatus.527 = INTEGER: up(1)
```

IF-MIB::ifDescr is a human-friendly abbreviation for the underlying OID – .1.3.6.1.2.1.2. Not every branch of the MIB has a human-friendly representation. In contrast, the equivalent subtree path with OpenConfig is /interfaces/interface[name='ge-0/0/0']/state/oper-status/. It doesn't take a leap in logic to understand the intent even if the operator is initially unfamiliar with this X-Path syntax.

Enabling OpenConfig With gRPC Transport

Enabling gRPC transport is trivial for testing purposes. Enabling it for production does overwhelmingly imply certificate-based authentication. As a result, it requires a functional public key infrastructure (PKI), specifically a certificate authority (CA), that can issue digital certificates.

CAUTION This book aspires to apply best practices. As a result, the telemetry configuration will be secure. Even so, maintaining and hardening a PKI environment is beyond the scope of this book. An insecure version will also be provided with an abundance of caution to not deploy in practice.

There are six steps needed to enable and secure gRPC:

1. Instantiate a root CA. This will need to be done on an x86 host. The example below uses OpenSSL to create a private key (.key extension) and self-signed certificate (.crt extension).

```
% openssl genrsa -out ca/root_ca.key 2048
% openssl req -x509 -new -key CA/root_ca.key -days 3650 -out CA/root_ca.crt -subj '/CN=Root CA'
```

2. You will next generate a private key and certificate signing request (CSR, .csr extension) for the gRPC client. In our example, jtimon is the gRPC client. The x86 host it runs on needs to present its identity to the gRPC servers (routers) for mutual authentication. The previously created CA can sign off on the CSR, generating a certificate for jtimon to use.

```
% openssl genrsa -out client/client.key 2048
% openssl req -new -key client/client.key -out client/client.csr -subj '/CN=localhost'
% openssl x509 -req -in client/client.csr -CA CA/root_ca.crt -CAkey CA/root_ca.key -CAcreateserial -out client/client.crt -days 365
```

3. Each router needs to generate a private key and CSR. The CA will then fulfill each CSR, as in the previous step, generating a certificate for the respective router. Both the certificate and private key are bundled together (.pem extension). The example below is for pe1.nyc:

```
% openssl genrsa -out router/pe1.nyc.key 2048
% openssl req -new -key router/pe1.nyc.key -out router/pe1.nyc.csr '/CN=pe1.nyc'
% openssl x509 -req -in router/router.csr -CA CA/RootCA.crt -CAkey CA/RootCA.key -CAcreateserial -out router/router.crt -days 365
% cat router/pe1.nyc.crt router/pe1.nyc.key > router/pe1.nyc.pem
```

4. All three identities – the root CA's and the jtimon host's certificates, as well as the router's own bundle – should be copied to the `/var/tmp` location on each router. As an example, this is what results on `pe1.nyc`:

```
user@pe1.nyc> file list /var/tmp/*.crt
/var/tmp/client.crt
/var/tmp/root_ca.crt

user@pe1.nyc> file list /var/tmp/*.pem
/var/tmp/pe1.nyc.pem
```

5. The certificates then need to be activated via configuration. The example below is for `pe1.nyc`:

```
[edit]
# set security pki ca-profile root_ca ca-identity root_ca
# set security certificates local local-cert load-key-file /var/tmp/pe1.nyc.pem
# commit
# run request security pki ca-certificate load ca-profile root_ca filename /var/tmp/root_ca.crt
```

6. Finally, gRPC can be enabled to listen on a secure port and require mutual authentication. A username and password will need to be provided by jtimon – the user needs to be present at the `[system login]` configuration hierarchy on the router:

```
user@pe1.nyc> show configuration
system {
  services {
    extension-service {
      request-response {
        grpc {
          ssl {
            port 32767;
            local-certificate local-cert;
            mutual-authentication {
              certificate-authority root_ca;
              client-certificate-request require-certificate-and-verify;
```

Strictly for testing purposes, you may wish to use gRPC insecurely. Do so outside a lab environment at your own peril. Skip steps 1-5 above and replace the stanza in step 6 with:

```
user@pe1.nyc> show configuration
system {
  services {
    extension-service {
      request-response {
        grpc {
          clear-text;
          skip-authentication;
```

SR Telemetry

With this backdrop, let's explore which SR statistics can be streamed. The most fundamental is the volume of segment routed traffic transiting an interface. During a migration from other MPLS transport protocols, this sensor can differentiate how much of the forwarding plane has shifted away to SR.

You can configure this as follows:

```
user@pe1.nyc> show configuration
protocols {
  isis {
    source-packet-routing {
      sensor-based-stats {
        per-interface-per-member-link ingress egress;
```

This installs sensors for each interface that has an SR next-hop.

Management plane: Verify the OpenConfig sensors have been installed

```
user@pe1.nyc> show isis spring sensor info
@IS-IS SENSOR INFORMATION

Per-interface-per-member-link Ingress Sensor:
-----
Sensor-name          Sensor-id
aggr_ingress_intf_sensor  3221225555

Per-interface-per-member-link Egress Sensor:
-----
Sensor-name          Sensor-id
ge-0/0/1.0           3221225556
ge-0/0/2.0           3221225557
```

```
user@pe1.nyc> show route table inet.3 128.49.106.0/32 detail

inet.3: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
128.49.106.0/32 (1 entry, 1 announced)
  *L-ISIS Preference: 14
  ...
  Next hop: 192.0.2.5 via ge-0/0/1.0
  Label operation: Push 1000
  ...
  Statistics ID Group: Kernel ID = 2, Stats IDs = { 3221225556 }
  Next hop: 192.0.2.7 via ge-0/0/2.0, selected
  Label operation: Push 1000
  ...
  Statistics ID Group: Kernel ID = 1, Stats IDs = { 3221225557 }
```

You can now subscribe to the OpenConfig path that exports these statistics (/mpls/signaling-protocols/segment-routing/interfaces/). Configure jtimon using a file similar to the one below, replacing the authentication credentials and paths to the private key and certificates as needed:

```
% cat dev/telemetry/pe1.nyc.ssl.json
{
  "host": "pe1.nyc",
  "port": 32767,
  "user": "bit",
  "password": "## SECRET-DATA ##",
  "cid": "client.jitmon",
  "tls": {
    "clientcrt": "dev/vault/client/client.crt",
    "clientkey": "dev/vault/client/client.key",
    "ca": "dev/vault/ca/root_ca.crt",
    "servername": "pe1.nyc"
  },
  "paths": [
    {
      "path": "/mpls/signaling-protocols/segment-routing/interfaces/",
      "freq": 10000
    }
  ]
}
```

The configuration is self-explanatory. The router is requested to publish the SR interface statistics every 10 seconds. Launch jitmon and observe what pe1.nyc reports:

```
% jitmon-darwin-amd64 --config dev/telemetry/pe1.nyc.ssl.json -print
Connecting to pe1.nyc:32767
...
Receiving telemetry data from pe1.nyc:32767
system_id: pe1.nyc
component_id: 0
sub_component_id: 0
path: sensor_1019_1://junos/services/segment-routing/interface/egress/usage:/mpls/signaling-
protocols/segment-routing/interfaces/:PFE
sequence_number: 0
timestamp: 1550427772636
sync_response: false
  key: __timestamp__
  uint_value: 1550427772637
  key: __prefix__
  str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/1.0']/
state/counters[name='oc-84']/
  key: out-pkts
  int_value: 5
  key: out-octets
  int_value: 420
```

As an ingress PE, you can see that inbound IP traffic from CEs has been sent out with an SR label via the interface connected to p1.nyc. It doesn't explain the traffic's destination as the imposed label isn't communicated. Note that the counter name (oc-84) uses an arbitrary number, unrelated to labels in use.

Pointing jitmon at p1.nyc shows bidirectional traffic flow. An equal number of packets ingress and egress ge-0/0/2 and ge-0/0/3 – these interfaces connect to pe1.nyc and pe2.nyc, respectively:

```

system_id: p1.nyc
...
str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/2.0']/
state/counters[name='oc-82']/
key: out-pkts
int_value: 5
key: out-octets
int_value: 440
...
str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/3.0']/
state/counters[name='oc-83']/
key: out-pkts
int_value: 5
key: out-octets
int_value: 440
system_id: p1.nyc
...
str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/2.0']/
key: in-pkts
int_value: 5
key: in-octets
int_value: 440
...
str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/3.0']/
key: in-pkts
int_value: 5
key: in-octets
int_value: 440

```

As a final step, instrumenting pe2.nyc shows SR traffic egressing ge-0/0/1, which is pe2.nyc's local interface to p1.nyc:

```

system_id: pe2.nyc
...
str_value: /mpls/signalling-protocols/segment-routing/interfaces/interface[name='ge-0/0/1.0']/
state/counters[name='oc-45']/
key: out-pkts
int_value: 5
key: out-octets
int_value: 420

```

The eagle-eyed will have observed that p1.nyc reports an average packet size 4 bytes larger than the PE routers – confirming the imposition of a single node SID derived MPLS label.

This is a good start, and we can only get more granular. Per-SID statistics, especially with a uniform domain-wide SRGB, can be enormously useful in calculating traffic matrices. The configuration to enable that is:

```

user@pe1.nyc> show configuration protocols isis | display inheritance no-comments
source-packet-routing {
  sensor-based-stats {
    per-interface-per-member-link ingress egress;
    per-sid ingress egress;
  }
}

```

Management plane: Verify additional OpenConfig sensors have been installed per-SID

```
user@pe1.nyc> show isis spring sensor info
@IS-IS SENSOR INFORMATION
```

```
Per-interface-per-member-link Ingress Sensor:
```

Sensor-name	Sensor-id
aggr_ingress_intf_sensor	3221225555

```
Per-interface-per-member-link Egress Sensor:
```

Sensor-name	Sensor-id
ge-0/0/1.0	3221225556
ge-0/0/2.0	3221225557

```
Per-sid Ingress Sensor:
```

Sensor-name	Sensor-id
17	3221225558
18	3221225559
19	3221225560
20	3221225561
1060	3221225562
1062	3221225563
1063	3221225564
1068	3221225565
1069	3221225566
1070	3221225567
1071	3221225568
1073	3221225569
1000	3221225570
1002	3221225571
1003	3221225572
1008	3221225573
1009	3221225574
1010	3221225575
1011	3221225576
1013	3221225577

```
IPv4/IPv6 Per-sid Egress Sensor:
```

Sensor-name	Sensor-id
L-ISIS-128.49.106.0	3221225578
L-ISIS-128.49.106.2	3221225579
L-ISIS-128.49.106.3	3221225580
L-ISIS-128.49.106.8	3221225581
L-ISIS-128.49.106.9	3221225582
L-ISIS-128.49.106.10	3221225583
L-ISIS-128.49.106.11	3221225584
L-ISIS-128.49.106.13	3221225585
L-ISIS-2001:db8::128:49:106:0	3221225586
L-ISIS-2001:db8::128:49:106:2	3221225587
L-ISIS-2001:db8::128:49:106:3	3221225588
L-ISIS-2001:db8::128:49:106:8	3221225589
L-ISIS-2001:db8::128:49:106:9	3221225590
L-ISIS-2001:db8::128:49:106:10	3221225591
L-ISIS-2001:db8::128:49:106:11	3221225592
L-ISIS-2001:db8::128:49:106:13	3221225593

An array of additional sensors has been created. The ingress per-SID sensor names correspond to the IPv4 and IPv6 node as well as adjacency SIDs. In the egress direction, they are named based on the underlying protocol and advertised prefix SID.

Now change the path that jtimon subscribes to: `/mpls/signaling-protocols/segment-routing/aggregate-sid-counters/`. The detail wells up from `pe1.nyc`, `p1.nyc`, and `pe2.nyc` to include per-SID traffic volume.

Management plane: Per-SID statistics at `pe1.nyc`

```
str_value: /mpls/signalling-protocols/segment-routing/aggregate-sid-counter[ip-addr='L-ISIS-128.49.106.0']/state/counters[name='oc-106']/out-pkts/
key: out-pkts
int_value: 5
key: out-octets
int_value: 420
```

Management plane: Per-SID statistics at `p1.nyc`

```
str_value: /mpls/signalling-protocols/segment-routing/aggregate-sid-counters/aggregate-sid-counter[mpls-label='1000']/state/counters[name='oc-95']/
key: in-pkts
int_value: 5
key: in-octets
int_value: 440
...
str_value: /mpls/signalling-protocols/segment-routing/aggregate-sid-counters/aggregate-sid-counter[mpls-label='1001']/state/counters[name='oc-96']/
key: in-pkts
int_value: 5
key: in-octets
int_value: 440
```

Management plane: Per-SID statistics at `pe2.nyc`

```
str_value: /mpls/signalling-protocols/segment-routing/aggregate-sid-counters/aggregate-sid-counter[ip-addr='L-ISIS-128.49.106.1']/state/counters[name='oc-67']/out-pkts/
key: out-pkts
int_value: 5
key: out-octets
int_value: 420
```

What is fantastic about this is that it shows unerringly how much traffic is destined from `pe1.nyc` to `pe2.nyc`, and vice versa. Observing `p1.nyc`'s counters shows how much traffic is destined to `pe1.nyc` or `pe2.nyc` from other PEs. With a universal SRGB, each router shares a common understanding of the label associated with a node SID. By summing counters associated with the same MPLS label, the operator can see how much traffic is being sent to the node SID originator through each transit router. Accumulating counters associated with a prefix SID shows the amount of traffic sent to a remote router. This can greatly simplify the construction of traffic matrices.

SR-TE Telemetry

Nowhere is telemetry more important than Segment Routing Traffic Engineering (SR-TE). Gleaning the volume of traffic directed into an SR-TE policy – loosely speaking, an SR-TE LSP – starts the feedback loop that can be used by a centralized control mechanism that can reroute the LSP based on bandwidth needs and availability.

To focus on its telemetry aspect, an SR-TE LSP has been created between pe2.nyc and pe1.iad. It is a *colored* LSP, which means it has been assigned an arbitrary, locally-significant numeric value. When a service route is learned via BGP, and carries an extended color community with a matching value, that route's protocol next hop is automatically pointed at this SR-TE LSP.

NOTE SR-TE is covered more extensively in Chapter 4.

Traffic destined to service routes that resolve over the colored LSP is accounted for as SR-TE statistics. As earlier, SR-TE telemetry is specifically enabled via:

```
user@pe2.nyc> show configuration
protocols {
  source-packet-routing {
    telemetry {
      statistics;
```

A simple, colored SR-TE LSP configuration is in effect:

```
user@pe2.nyc> show configuration
protocols {
  source-packet-routing {
    lsp-external-controller pccd;
    segment-list pe2.iad_v6_path0 {
      hop1 label 24;
      hop2 {
        label 1061;
        label-type {
          node;
        }
      }
      hop3 {
        label 1071;
        label-type {
          node;
        }
      }
    }
  }
  source-routing-path pe2.nyc:pe1.iad_v6 {
    to 2001:db8::128:49:106:11;
    color 7100;
    binding-sid 1000000;
    primary {
      pe2.iad_v6_path0;
    }
  }
}
```

Management plane: confirm sensor is attached to SR-TE LSP

```

user@pe2.nyc> show route table inet6color.0 detail
inet6color.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
2001:db8::128:49:106:11-7100<c6>/160 (1 entry, 1 announced)
    *SPRING-TE Preference: 8
    ...
    Next hop: fe80::5668:a3ff:fe1e:6af via ge-0/0/2.0 weight 0x1, selected
    ...
    Protocol next hop: 24
    Label operation: Push 1071, Push 1061(top)
    ...
    SRTE Policy State:
        SR Preference/Override: 100/100
        BSID: 1000000
    SRTE Policy Statistics State:
    Statistics Enabled: Ingress, Transit
        Ingress Sensor Status: Active, Sensor Id: 3758096403

```

Control plane: confirm service route points at SR-TE LSP

```

user@pe2.nyc> show route 2001:db8::198:51:100:54 detail

inet6.0: 24 destinations, 31 routes (24 active, 0 holddown, 0 hidden)
2001:db8::198:51:100:54/127 (2 entries, 1 announced)
    *BGP Preference: 170/-101
    ...
    Next hop: fe80::5668:a3ff:fe1e:6af via ge-0/0/2.0 weight 0x1, selected
    ...
    Protocol next hop: 2001:db8::128:49:106:11-7100<c6>
    ...
    Communities: color:0:7100

```

By default, there are two sensors attached to an SR-TE LSP. The ingress sensor measures the amount of IP traffic directed into the policy, and the transit sensor measures the amount of labeled traffic similarly directed into the policy. The transit sensor is necessary if the *binding SID* (bSID), another locally-significant and separate from the color numeric identifier, associated with the SR-TE policy is used to steer MPLS-labeled traffic into the LSP.

As with all segment routing sensors, nothing is exported until non-zero statistics accumulate. As soon as IP traffic matching the colored service route arrives, the counters increase and are subsequently published at the frequency requested by the gRPC client.

Management plane: SR-TE IP statistics

```

system_id: pe2.nyc
component_id: 0
sub_component_id: 0
path: sensor_1005_1:/junos/services/segment-routing/traffic-engineering/ingress/usage/;/mpls/
signaling-protocols/segment-routing/sr-te-ip-policies/;PFE
sequence_number: 17
timestamp: 1550946298603
sync_response: false

```

```

key: __timestamp__
uint_value: 1550946298604
key: __prefix__
str_value: /mpls/signalling-protocols/segment-routing/sr-te-ip-policies/sr-te-ip-policy[to-
address='2001:db8::128:49:106:11', color='7100']/state/counters[name='oc-19']/
key: packets
int_value: 5
key: bytes
int_value: 280

```

The IP policy name, intuitively enough, contains the endpoint of the LSP as well as its color. If MPLS labeled traffic had been forwarded into this LSP, using the binding SID, the policy name would additionally contain the bSID identifier. Also inarguable is SR's treatment of IPv6 as a first-class citizen throughout the entire stack – from creating a colored SR-TE policy towards an IPv6 endpoint, to directing IPv6 traffic into that LSP, ending with receiving telemetry for the effort.

SR EPE Telemetry

The last application of SR telemetry is with *egress peer engineering* (EPE). Broadly speaking, an autonomous system boundary router (ASBR) can associate a label with a BGP peer, a specific link to a given BGP peer, or a set of BGP peers. This label is only installed in that ASBR's forwarding plane, and is exported to a controller via BGP-LS as an EPE SID. The controller can craft SR-TE policies that lead beyond a particular ASBR to a distinct peer, a unique link to one peer, or even a collection of peers.

As used earlier, an SR-TE LSP has been installed on pe2.nyc to a particular BGP peer of pe1.iad. This requires the EPE label configured at pe1.iad to be present at the bottom of the segment list in the SR-TE policy. SR-EPE telemetry needs to be explicitly enabled, as you would have come to expect:

```

user@pe1.iad> show configuration
protocols {
    bgp {
        egress-te-sid-stats;
        group PEERS {
            neighbor 198.51.100.56 {
                egress-te-node-segment {
                    label 1044444;
                }
            }
        }
    }
}

user@pe2.nyc> show configuration
protocols {
    source-packet-routing {
        lsp-external-controller pccd;
        segment-list pe1.iad_v4_path1_epe {
            hop1 label 21;
            hop2 {
                label 1011;
                label-type {
                    node;
                }
            }
        }
    }
}

```

```

    }
  }
  hop3 label 1044444;
}
source-routing-path pe2.nyc:pe1.iad_v4 {
  to 128.49.106.11;
  color 7100;
  binding-sid 1000001;
  primary {
    pe1.iad_v4_path1_epe;
  }
}

```

Control plane: confirm sensor is installed for the EPE SID

```
user@pe1.iad> show route label 1044444 detail
```

```

mpls.0: 36 destinations, 36 routes (36 active, 0 holddown, 0 hidden)
1044444 (1 entry, 1 announced)
  *BGP-LS-EPE Preference: 170
  ...
  Next hop: 198.51.100.56 via ge-0/0/12.0, selected
  Label operation: Pop
  ...
  Stats ID Group: Kernel ID = 39, Stats IDs = { 268435458 }

```

Traffic matching the colored service route is transmitted, resulting in pe1.iad emitting telemetry for the EPE SID:

```

system_id: pe1.iad
component_id: 0
sub_component_id: 0
path: sensor_1006_2:/junos/services/segment-routing/sid/usage:/mpls/signaling-protocols/segment-
routing/aggregate-sid-counters/:PFE
sequence_number: 895
timestamp: 1550958775682
sync_response: false
  key: __timestamp__
  uint_value: 1550958775684
  key: __prefix__
  str_value: /mpls/signalling-protocols/segment-routing/aggregate-sid-counters/aggregate-sid-
counter[mpls-label='1044444']/state/counters[name='oc-2']/
  key: in-pkts
  int_value: 5
  key: in-octets
  int_value: 440

```

OAM (Operations, Administration, Maintenance)

Admittedly, OAM is often an afterthought for network engineers. It's only when things tilt sideways in production that many reach for tools that assert not just control plane path liveness, but also verify data plane hygiene.

This is even more important for SR-TE as it eliminates path signaling. Instead, it delegates this to sBFD (Seamless Bidirectional Forwarding Detection). BFD has

been a workhorse for detecting data path failures. Each node in a point-to-point relationship starts by learning the other's discriminator, the node and application identifier in BFD-speak. Following this discovery, a BFD session establishes for connectivity checking.

sBFD simplifies BFD by eliminating the handshake needed to learn discriminators. Instead, the operator assigns at least one local discriminator to each node in the network. In order to establish sBFD for an SR-TE path, the head-end is configured with the remote node's discriminator.

The head-end acts as the initiator and pushes the SR-TE path's label stack onto the liveness checking packets and ensuring the packets traverse the same path that user traffic will; the SR-TE policy destination statelessly responds to the sBFD packets. The SR-TE path is viable as long as the sBFD session is up at the ingress router.

You can configure sBFD by adding a discriminator on each node in the network. Let's add sBFD liveness checking to the IPv4 SR-TE path we have defined from pe2.nyc to pe1.iad:

```
user@pe2.nyc> show configuration
protocols {
  bfd {
    sbfd {
      local-discriminator 100;
    }
  }
  ...
  source-packet-routing {
    source-routing-path pe2.nyc:pe1.iad_v4 {
      primary {
        pe1.iad_v4_path0 {
          bfd-liveness-detection {
            sbfd {
              remote-discriminator 111;
            }
            minimum-interval 100;
            multiplier 4;
          }
        }
      }
    }
  }
}

user@pe1.iad> show configuration
protocols {
  bfd {
    sbfd {
      local-discriminator 111;
    }
  }
}
```

After adding this configuration the SR-TE path will remain Up as long as sBFD forwarding exists between the two routers, using the same label stack.

Control plane: Verify pe2.nyc's sBFD session to pe1.iad is Up

```
user@pe2.nyc> show bfd session extensive
```

Address	State	Interface	Detect Time	Transmit Interval	Multiplier
127.0.0.1	Up		0.400	0.100	4

Client SPRING-TE, TX interval 0.100, RX interval 0.100
 Session up time 01:41:41
 Local diagnostic None, remote diagnostic None

```

Remote state Up, version 1
Session type: Multi hop BFD (Seamless)
Min async interval 0.100, min slow interval 1.000
Adaptive async TX interval 0.100, RX interval 0.100
Local min TX interval 0.100, minimum RX interval 0.100, multiplier 4
Remote min TX interval 0.100, min RX interval 0.100, multiplier 4
Local discriminator 18, remote discriminator 111
Echo mode disabled/inactive
Remote is control-plane independent
Path-Name V4-pe1.iad_v4_path0
Session ID: 0x0

```

1 sessions, 1 clients

Cumulative transmit rate 10.0 pps, cumulative receive rate 10.0 pps

Control plane: Verify pe2.nyc's SR-TE policy has a valid path

```

user@pe2.nyc> show spring-traffic-engineering lsp detail name pe2.nyc:pe1.iad_v4
Name: pe2.nyc:pe1.iad_v4
Tunnel-source: Static configuration
To: 128.49.106.11-7100<c>
State: Up
  Path: pe1.iad_v4_path0
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
BFD status: Up BFD name: V4-pe1.iad_v4_path0
  SR-ER0 hop count: 2
    Hop 1 (Strict):
      NAI: None
      SID type: 20-bit label, Value: 21
    Hop 2 (Loose):
      NAI: None
      SID type: 20-bit label, Value: 1011

```

The first hop uses the adjacency SID to p1.nyc (21). If the adjacency between pe2.nyc and p1.nyc falters, this SR-TE policy will be brought Down.

Control plane: Verify the SR-TE policy goes Down if the label stack is rendered unusable

```

user@pe2.nyc> show isis adjacency

```

Interface	System	L State	Hold (secs)	SNPA
ge-0/0/1.0	p1.nyc	1 Down	0	
ge-0/0/2.0	p2.nyc	1 Up	25	

```

user@pe2.nyc> show spring-traffic-engineering lsp detail name pe2.nyc:pe1.iad_v4
Name: pe2.nyc:pe1.iad_v4
Tunnel-source: Static configuration
To: 128.49.106.11-7100<c>
State: Down
  Path: pe1.iad_v4_path0
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
BFD status: Down BFD name: V4-pe1.iad_v4_path0
  SR-ER0 hop count: 2
    Hop 1 (Strict):
      NAI: None

```

```

SID type: 20-bit label, Value: 21
Hop 2 (Loose):
NAI: None
SID type: 20-bit label, Value: 1011

```

There are no alternate SID lists, otherwise known as paths, for the policy to utilize. User traffic will resolve over a non-SR-TE path, if one is available. In contrast, statically configured SR-TE paths that don't take advantage of sBFD remain misleadingly Up.

Debuggability

Telemetry is but one part of observability. Troubleshooting network performance and availability depends on both the clarity of the underlying protocols and the diagnostics and logging of the available implementations.

SR's minimization of state on transit routers, and its use of an uniform SRGB, make it easier for a human to parse node SID label forwarding actions – there is less to grok. Junos builds on this simplicity with a rich logging functionality to detect human error.

The SRMS section gave an example of how misconfigured node SIDs were flagged. There are two different types of conflicts that SR identifies:

SID conflict: this is caught if different SIDs are assigned to the same prefix.

Prefix conflict: this occurs if different prefixes are assigned the same SID.

The earlier example demonstrated a SID conflict. Let's temporarily force a prefix conflict and use gRPC to stream that as an event. Traditional syslog remains available of course, but just like SNMP being succeeded by gRPC for telemetry, events can now be transmitted over gRPC as well.

Let's configure jtimon to subscribe to `/junos/events/`. To cause a prefix conflict, `pe1.iad` is temporarily configured with an additional loopback address that actually belongs to `p1.iad` (128.49.106.9/32). This would be fine if these addresses were used to generate a common anycast SIDs. Instead, `pe1.iad` is misconfigured to advertise a different prefix index (99) than `p1.iad` (9) using an IS-IS export policy.

Control plane: `pe1.iad` misconfiguration to cause a prefix conflict

```

policy-statement ISIS-EXPORT {
  term PREFIX-SID-CONFLICT {
    from {
      protocol direct;
      interface lo0.0;
      route-filter 128.49.106.9/32 exact;
    }
  }
}

```

```

    then {
        prefix-segment index 99;
        accept;
    ...

```

```

user@pe1.iad> show configuration protocols isis | display inheritance no-comments
...
export ISIS-EXPORT;

```

Management plane: streamed system event about the prefix conflict

```

system_id: pe1.iad
component_id: 65535
sub_component_id: 0
path: sensor_1004:/junos/events:/junos/events:/event
sequence_number: 5831
timestamp: 1551326098693
sync_response: false
  key: __timestamp__
  uint_value: 1551326098694
  key: __junos_re_stream_creation_timestamp__
  uint_value: 1551326098693
  key: __junos_re_payload_get_timestamp__
  uint_value: 1551326098693
  key: __junos_re_event_timestamp__
  uint_value: 1551326098693
  key: __prefix__
  str_value: /junos/events/event[id='RPD_ISIS_PREFIX_SID_CNFLCT' and type='2' and facility='3']/
  key: timestamp/seconds
  uint_value: 1551326098
  key: timestamp/microseconds
  uint_value: 692834
  key: priority
  uint_value: 3
  key: pid
  uint_value: 3211
  key: message
  str_value: RPD_ISIS_PREFIX_SID_CNFLCT: IS-
IS detected L1 prefix segment index '9' originated by 'p1.
iad' for prefix 128.49.106.9/32 conflicting with self-
originated L1 prefix segment index '99' for prefix 128.49.106.9/32
  key: daemon
  str_value: rpd
  key: hostname
  str_value: pe1.iad
  key: __prefix__
  str_value: /junos/events/event[id='RPD_ISIS_PREFIX_SID_CNFLCT' and type='2' and facility='3']/
  attributes[key='isis-level']/
  key: value
  str_value: 1
  key: __prefix__
  str_value: /junos/events/event[id='RPD_ISIS_PREFIX_SID_CNFLCT' and type='2' and facility='3']/
  key: logoptions
  int_value: 9

```

Attention is beckoned and the misconfiguration recitified. SR specifies a number of tie-breaking rules to address the conflict until human intervention takes place. The rules are evolving. At this point they are:

1. Prefer IPv6 entries over IPv4
2. Prefer longest prefix lengths
3. Prefer the numerically lowest prefix address
4. Prefer the numerically lowest segment index

Prefix conflicts are resolved first; SID conflict resolution follows on the outstanding entries. With this, let's surface from our dive into observability and move on to optimizability.

Chapter 4

Optimizability

The previous chapters have looked at various use cases for introducing segment routing. This chapter looks specifically at the traffic engineering (TE) aspects of SR. In particular it describes:

- The TE process model
- The role of explicit path definition
- Routing constraints and several salient properties of SR SID types
- Distributed and centralized path computation
- And finally, an end-to-end SR-TE solution for the sample network leveraging a centralized controller.

The TE Process Model

In the TE process model provided in Figure 4.1, the operator, or a suitable automaton, acts as a “controller” in an adaptive feedback control system. This system includes:

- A set of interconnected network elements (IP/MPLS network)
- A network state/topology monitoring element
- A network performance monitoring element
- A network configuration management element

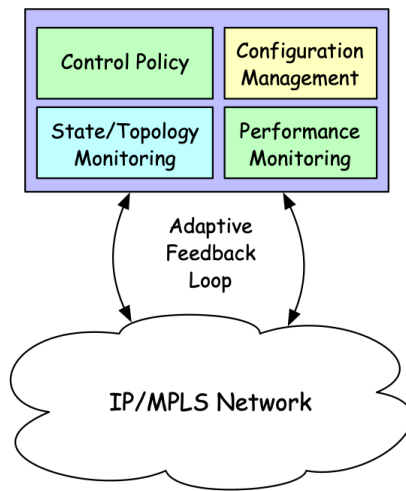


Figure 4.1 TE Process Model

The operator/automaton formulates a control policy, observes the state of the network through the monitoring system, characterizes the traffic, and applies control actions to drive the network to an optimal state. This can be done reactively, based on the current state of the network, or proactively, based on a predicted future state.

The adaptive feedback loop may be implemented in each node in the network, i.e. distributed TE, or in a centralized manner, i.e. using a Path Computation Element (PCE). Furthermore, various hybrid models are possible.

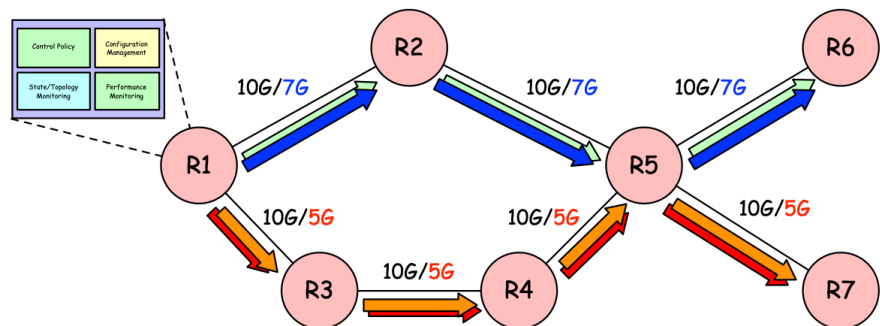


Figure 4.2 Distributed Versus Centralized TE Control-loop

Explicit Routing: A Tool for TE

Explicit routing may be desired to optimize network resources or provide very strict service guarantees but it is not by itself a TE solution. The result of the TE process model, the important part of a traffic engineering solution, likely requires the programming of an explicit route to program the resulting computation. It is therefore desired to be able to define a strict path across a network for one or more LSPs. Both RSVP-TE and SR provide a means to explicitly define strict paths (strict hops, loose hops and/or abstract/anycast hops) across a network.

Explicit Routing with RSVP

Network operators request, typically through configuration, LSPs that meet specific constraints. For example, a network operator could request an LSP that originates at Node R1, terminates at Node R6, reserves 100 megabits per second, and traverses blue interfaces only. A path computation module, located on a central controller – such as the PCE – or on the ingress router, computes a path that satisfies all of the constraints. Figure 4.3 illustrates the resulting RSVP path setup and reserve messaging to set up the LSP and the resulting MPLS forwarding table label operations.

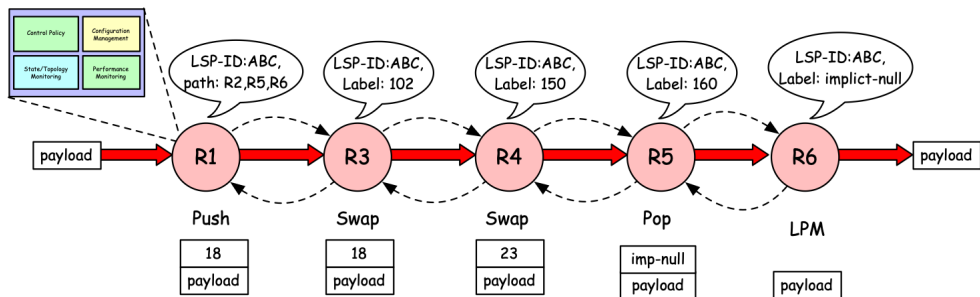


Figure 4.3

RSVP Signaling and Labels Allocation for an Explicit Tunnel

The following configuration example illustrates the required configuration. It's worth noting that there are no routing constraints defined for the LSP. In other words, the only configuration shown is the definition of the explicit path. In many (or most) RSVP-TE deployments the paths are not explicitly defined but instead a routing constraint is defined, such as bandwidth or link affinity, such that the ingress node or external CSPF computes the explicit path dynamically to meet the routing constraint.

R1 configuration

```

protocols {
  mpls {
    label-switched-path te-lsp-to-r6 {
      to 1.1.1.6;
      primary {
        explicit-path-to-r6;
      }
      ...
    }
    path explicit-path-to-r6 {
      10.1.13.2 strict;
      10.1.34.2 strict;
      10.1.45.2 strict;
      10.1.56.2 strict;
    }
  }
}

```

Explicit Path Definition with SR

Much like RSVP-TE explicit LSPs, a network operator will request, typically through configuration, LSPs that meet specific constraints. The main difference is not only the specific configuration syntax, though they are quite similar, but the concept of describing the ‘routing constraint’ using the same CLI constructs. To provide complex TE constraints, an ingress SR node may need to rely on an external controller or PCE.

Using the same example as in the RSVP-TE example, a network operator could request an LSP that originates at Node R1, terminates at Node R6, reserves 100 megabits per second, and traverses blue interfaces only but would require an external controller/PCE to compute the path. An external PCE is required only because of the specified bandwidth constraint as SR doesn’t signal its reservations. If bandwidth is not required, a distributed path computation completed by R1 would suffice for the SR path computation. Figure 4.4 illustrates the resulting SR path and MPLS forwarding table label operations. Note that there is a fairly significant difference in the label forwarding operations for a SR-TE LSP compared to the SPF SR LSP and the previous RSVP-TE LSPs.

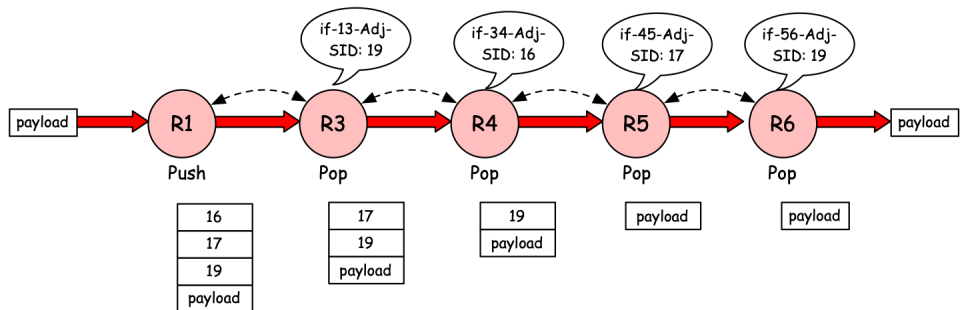


Figure 4.4

SR Signaling and Labels Allocations for an Explicit Tunnel

SR adjacency SIDs are dynamically allocated by default. Because the next configuration example illustrates using traditional CLI techniques for describing the explicit path, where each hop in the path is specified by its label/SID, it is recommended that labels are pre-planned and statically assigned so that each link has a unique label/SID, much like how IP addressing is handled. It's worth noting that an SR-TE path can also be described, via CLI, using traditional IP addresses as the specified hops in the path and allowing the ingress router to resolve the SID and label stack.

R1: defining static adjacency SIDs

```
protocols {
  isis {
    interface ge-0/0/1.0 {
      level 2 {
        ipv4-adjacency-segment {
          unprotected label 19;
        }
      }
    }
  }
}
```

R1: defining explicit SR tunnels

```
protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.6;
      primary {
        explicit-path-to-r6;
      }
      segment-list explicit-path-to-r6 {
        segment1 label 19;
        segment2 label 16;
        segment3 label 17;
        segment4 label 19;
      }
    }
  }
}
```

As mentioned above, you may use the `auto-translate` option and describe the path as a series of IP hops, like an RSVP-TE path, and Junos will translate the SIDs. This has a nice advantage of removing the need to statically configure adjacency SIDs as well, ensuring a controller, who may have calculated the path, is not required to change the path in the event of a link transitioning down.

```
protocols {
  source-packet-routing {
    segment-list explicit-path-to-r6 {
      hop1 ip-address 10.1.13.2;
      hop2 ip-address 10.1.34.2;
      hop3 ip-address 10.1.45.2;
      hop4 ip-address 10.1.56.2;
      auto-translate;
    }
  }
}
```

Loose Hops, Prefix-SIDs, and Anycast-SIDs

Segment Routing SIDs

A segment identifier (SID) identifies each segment. Network operators allocate SIDs using procedures that are similar to those used to allocate private IP (i.e., RFC 1918) addresses. Every SID maps to an MPLS label with SR-MPLS. SR-capable routers advertise the SIDs via the IGP. The IGP floods this data, in addition to the previously mentioned TE link attributes described above, throughout the IGP domain. Therefore, each node within the IGP domain maintains an identical copy of a link-state database (LSDB) and a traffic engineering database (TED). The following segment types are the most common and will be used in this chapter to describe several relevant use cases.

Adjacency: Adjacency segments represent an IGP adjacency between two routers. Junos allocates adjacency SIDs dynamically but they can also be statically configured. As previously mentioned this may be useful in some scenarios:

```
protocols {
  isis {
    interface ge-0/0/1.0 {
      level 2 {
        ipv4-adjacency-segment {
          unprotected label 19;
        }
      }
    }
  }
}
```

Prefix: Prefix segments represent the IGP least cost path between any router and a specified prefix. Prefix segments contain one or more router hops. A node SID is also a type of prefix SID:

```
policy-options {
  policy-statement prefix-sid {
    term 0 {
      from {
        route-filter 10.1.13.1/30 exact;
      }
      then {
        prefix-segment index 1000;
        accept;
      }
    }
  }
}

isis {
  export prefix-sid;
}
```

Anycast: Anycast segments are like prefix segments in that they represent the IGP least cost path between any router and a specified prefix. However, the specified prefix can be advertised from multiple points in the network. Note that in the example, the CLI shows only a single node announcing the anycast-SID. In an actual deployment all nodes that are part of the anycast group would advertise the same anycast-SID.

```

interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 1.1.1.4/32;
      }
      family iso {
        address 49.0001.0001.0004.00;
      }
    }
  }
}

policy-options {
  policy-statement anycast-sid {
    term 0 {
      from {
        route-filter 1.1.1.4/32 exact;
      }
      then {
        prefix-segment index 405;
        accept;
      }
    }
  }
}

protocols {
  isis {
    export anycast-sid;
  }
}

```

Binding: Binding prefixes represent tunnels in the SR domain. The tunnel can be another SR-Path, an LDP-signaled LSP, an RSVP-TE signaled LSP, or any other encapsulation:

```

protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.5;
      binding-sid 2000
      primary {
        explicit-path-to-r5;
      }
    }
  }
}

```

The Critical Role of Maximum SID Depth

Maximum SID depth (MSD) is a generic concept defining the number of SID's LSR hardware and software capable of imposing on a given node. It is defined in various IETF OSPF/ISIS/BGP-LS/PCEP drafts. When SR paths are computed, it is critical that the computing entity learn the MSD that can be imposed at each node or link for a given SR path. This ensures that the SID stack depth of a computed path does not exceed the number of SIDs the node is capable of imposing.

Setting, Reporting, and Advertising MSD

When using PCEP to communicate with a PCE for LSP state reporting, control, and provisioning, the MSD is reported. The following CLI can be used to increase the reported MSD value of 5:

```

protocols {
  pcep {
    pce pccd {
      max-sid-depth 16;
    }
  }
}

```

While this reports MSD to a controller via a control plane protocol, Junos also requires that ingress interfaces that will be imposing labels also have their label imposition increased from the default value of 3:

```

groups {
  if-group {
    interfaces {
      <ge-*> {
        unit <*> {
          family mpls {
            maximum-labels 16;
          }
        }
      }
    }
  }
}

apply-groups if-group;

```

SID Depth Reduction

Because MSD is such a critical concern in many SR scenarios, the idea that the end-to-end path need not be completely specified is a very popular idea. In the previous examples, the example CLI specified the individual hops of the SR-TE LSP (and the RSVP-TE LSP) and can be referred to as *strict*. A strict hop means that the specified LSR must be directly connected to the previous hop. A loose hop, on the other hand, means that the path must pass through the specified LSR, but the LSR does not have to be directly connected to the last hop; any valid route between the two can be used. Let's look at an example using the topology illustrated in Figure 4.5.

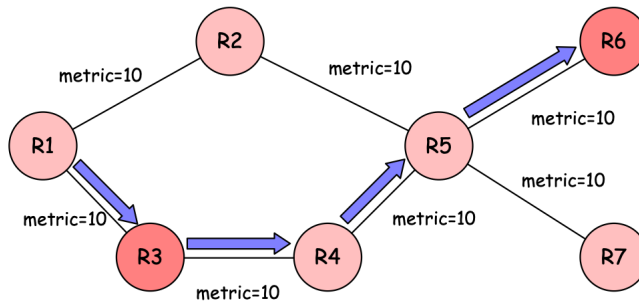


Figure 4.5 Example Topology for a Loose-hop LSP

Let's say you want an LSP to go from R1 to R6 via the lower route of R3, R4, and R5. To achieve this goal you merely need to specify R3 as the first hop in the path and then let normal routing take over from there since the path from R3 to R6 via

R5 has an IGP metric of 30 while the path via R1 has an IGP metric of 40. The resulting label stack for the SR-TE LSP is only two labels deep (node-SID for R3 and node-SID for R6) instead of four as previously seen:

```
protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.6;
      primary {
        explicit-path-to-r6;
      }
    }
  }
  segment-list explicit-path-to-r6 {
    hop1 ip-address 1.1.1.3 {
      label-type node;
    }
    hop2 ip-address 1.1.1.6 {
      label-type node;
    }
  }
  auto-translate;
}
```

You can equate this, and the previous examples, to the MPLS version of a static route. Like static routes, manually configured paths are useful when you want explicit control, but also like static routes, they pose an administrative problem when you need to establish many paths and/or want the network to dynamically react to new events such as a link coming Up or going Down.

Special care must be taken when explicitly configuring loose hops for a path, RSVP or SR. Link failures can result in unexpected forwarding behavior. For example, let's say that the link between R1 and R3 fails, as illustrated in Figure 4.6. Since the SR-TE LSP is a static instruction set and the node SID specified in the path is still reachable on the network, the resulting LSP's path will be suboptimal, yet valid.

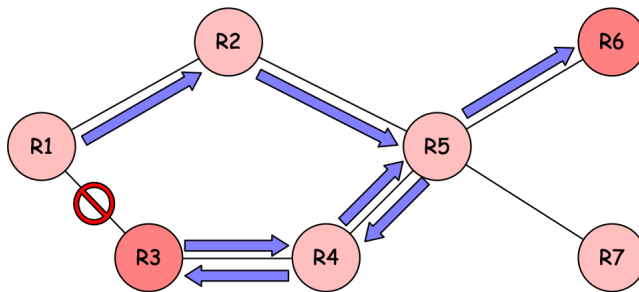


Figure 4.6

Suboptimal Forwarding in SR Networks

Binding SIDs represent another option for reducing the label stack depth when configuring explicit paths, as shown in Figure 4.7. Using the same simple example topology, an SR-TE LSP could be created from R3 to R5 and advertised with a binding-SID such that the ingress LSP from R1 to R6 referenced the binding SID in its path and the resulting label stack is only two deep.

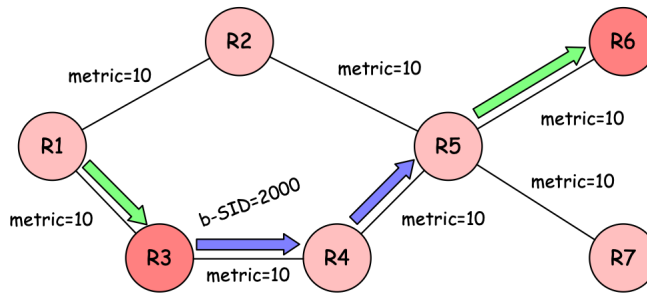


Figure 4.7 Label Stack Reduction Using Binding SIDs

R3 to R5 SR-TE LSP with binding-SID = 2000

```

protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.5;
      binding-sid 1000002
      primary {
        explicit-path-to-r5;
      }
    }
  }
  segment-list explicit-path-to-r5 {
    hop1 label 43;
    hop2 label 54;
  }
}

```

And then the SR-TE LSP from R1 to R6 would look like:

```

protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.6;
      primary {
        explicit-path-to-r6;
      }
    }
  }
  segment-list explicit-path-to-r6 {
    hop1 label 1000002;
    hop2 label 1060;
  }
}

```

Anycast SIDs represent another SID type used to create some interesting forwarding behaviors in SR networks. Since multiple nodes, comprising the anycast group, announce the same prefix SID, an ingress or transit node will forward towards the closest, from an IGP metric perspective, node announcing the prefix SID. This enables an operator to introduce load balancing and high availability scenarios that are somewhat unique to SR networks. Again, using our simple network topology shown in Figure 4.8, let's assume that R4 and R5 announce the same anycast SID, 405. You can now create an ingress LSP from R1 to R6 that results in equal cost load balancing between the R1, R2, R5, R6 path and the R1, R3, R4, R5, and R6 paths as shown.

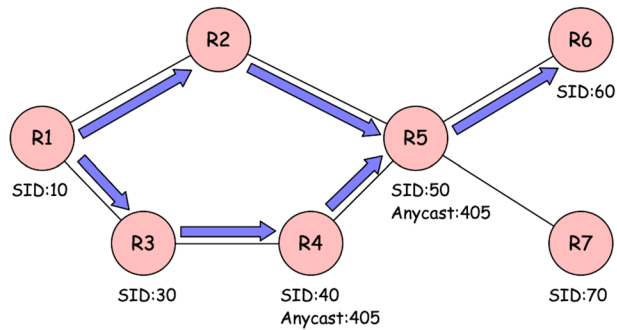


Figure 4.8 Load Balancing With Anycast SIDs

Again, while not the primary goal of using anycast SIDs, the label stack has been reduced by specifying the anycast SID as a hop in our SR-TE LSPs path:

```

protocols {
  source-packet-routing {
    source-routing-path sr-te-lsp-to-r6 {
      to 1.1.1.6;
      primary {
        explicit-path-to-r6;
      }
    }
    segment-list explicit-path-to-r6 {
      hop1 label 1405;
      hop2 label 1060;
    }
  }
}

```

Another way for the ingress LSR to determine the path is to dynamically calculate it. Just as OSPF and IS-IS use a Shortest Path First (SPF) algorithm to calculate a route, the ingress LSR can use a modification of the SPF algorithm called *Constrained Shortest Path First* (CSPF) to calculate a path. Let's explore dynamic path calculation next.

Dynamic Path Calculation and Routing Constraints

Thus far we have looked at various ways of defining a non-shortest path or explicit path using the various types of SIDs that SR offers along with a few special considerations. But defining and managing tens, hundreds, thousands, or tens of thousands of static SR paths is unscalable. We must therefore look back to the TE process model and explore how to define 'simple' routing constraints that describe how SR-TE LSPs paths should be computed. In other words, the explicit paths are not TE but rather a means to enable TE. First, let's look at information distribution or state/topology monitoring.

Link and Node Attributes and Routing Constraints

The most important requirement for TE is the dissemination of link and node characteristics. Just as SIDs are reliably flooded throughout a routing domain, link and node characteristics along with TE-oriented resource availability are also flooded throughout a TE domain using extensions to the IGP. Enabling the use of link-state routing protocols to efficiently propagate information pertaining to resource availability in their routing updates is achieved by the addition of extensions to the link-state routing protocol. Link-state routing protocols manage not only the flooding of updates in the network upon link-state or metric change, but also bandwidth availability from a TE perspective. These resource attributes are flooded by the set of routers in the network to make them available to head end routers for use in TE tunnel LSP path computation (dynamic tunnels).

Link-state announcements carry information lists that describe a given router's neighbors attached networks, network resource information, and other relevant information pertaining to the actual resource availability that might be later required to perform a constraint-based SPF calculation. OSPF and IS-IS have been provided with extensions to enable their use in an MPLS TE environment to propagate information pertaining to resource availability and in dynamic LSP path selection. These link and node attributes take the form of link colors, shared risk link group associations, available bandwidth, and metric types (TE or IGP) to name a few. These attributes are available in the TED to be used by the computing entity. Again, using our simple example topology shown in Figure 4.9, the following has been added to R5:

```
routing-options {
  srlg {
    common-254 {
      srlg-value 100;
    }
  }
}

protocols {
  mpls {
    admin-groups {
      red 23;
      blue 32;
    }
    interface ge-0/0/1.0 {
      srlg common-254;
      admin-group blue;
    }
    interface ge-0/0/3.0 {
      srlg common-254;
      admin-group red;
    }
  }
  isis {
    interface ge-0/0/3.0 {
      level 2 te-metric 100;
    }
  }
}
```

This results in a TE topology that looks like Figure 4.9 where the R5 to R2 link has been colored blue and added to the SRLG named `common-254` and the link from R5 to R4 has been colored red, given a `te-metric` of 100 and also added to the same SRLG.

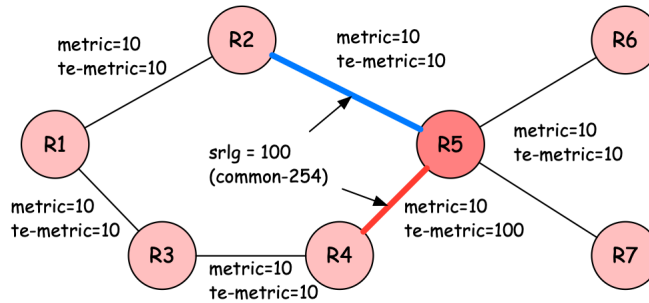


Figure 4.9

Link TE Attributes

Let's observe the contents of the TED for R5's link to R4 and see how the link TE information has been flooded on R1, using ISIS TE extensions, so that it can be used for path computation, and specifically for, constraint inclusion or exclusion:

```
user@R1> show ted database R5.00 extensive
To: R4.00(1.1.1.4), Local: 10.1.45.2, Remote: 10.1.45.1
Local interface index: 335, Remoteinterface index: 334
Color: 0x800000 red
Metric: 100
IGP metric: 10
Static BW: 1000Mbps
Reservable BW: 1000Mbps
Available BW [priority] bps:
  [0] 1000Mbps   [1] 1000Mbps   [2] 1000Mbps   [3] 1000Mbps
  [4] 1000Mbps   [5] 1000Mbps   [6] 1000Mbps   [7] 1000Mbps
Interface Switching Capability Descriptor(1):
  Switching type: Packet
  Encoding type: Packet
  Maximum LSP BW [priority] bps:
    [0] 1000Mbps   [1] 1000Mbps   [2] 1000Mbps   [3] 1000Mbps
    [4] 1000Mbps   [5] 1000Mbps   [6] 1000Mbps   [7] 1000Mbps
SRLGs: common-254
P2P Adjacency-SID:
  IPV4, SID: 299840, Flags: 0x30, Weight: 0
<output truncated>
```

Distributed SR Constraint-based SPF

In the normal SPF calculation process, a router places itself at the head of the tree with the shortest paths calculated to each of the destinations, taking only the least metric, or cost route, to the destination into account. In this calculation, a key concept to note is that no consideration is given to the bandwidth of the links on the other paths. If the attributes required for a given path include parameters beyond

simply the IGP cost or metric such as link color, the topology can be constrained to eliminate the links that do not allow for the mentioned requirements, such that the SPF algorithm, returns a path with both link cost and link inclusion/exclusion requirements.

With CSPF, you use more than the link cost to identify the probable paths that can be used for TE LSP paths. The decision of which path is chosen to set up a TE LSP path is performed at the computing entity, after ruling out all links that do not meet a certain criteria, such as link colors, in addition to the `te-cost` of the link. The result of the CSPF calculation is an ordered set of SIDs that map to the next-hop addresses of routers that form the TE LSP. Therefore, multiple TE LSPs could be instantiated by using CSPF to identify probable links in the network that meet the criteria.

Constraint-based SPF can use either administrative weights or TE metrics during the constraint-based computation. In the event of a tie, the path with the highest minimum bandwidth takes precedence, followed by the least number of hops along the path. If all else is equal, CSPF picks a path at random and chooses the same to be the TE LSP path of preference.

As previously mentioned, SR paths contain a few salient attributes, mainly MSD and ECMP attributes, resulting in the need for slightly different CSPF results than those of traditional RSVP-TE paths. As a result, Junos, as a distributed CSPF (we'll talk about external, centralized, CSPFs in a moment), has been enhanced to not only provide an ordered set of adjacency SIDs for a path, but also to minimize or meet the MSD for a given ingress router's MSD, as well as to leverage any available ECMPs along the resulting set of candidate paths by offering node SIDs within the segment list.

The SR-TE candidate paths will be locally computed such that they satisfy the configured routing constraints. The multi path CSPF computation results will be an ordered set of adjacency SIDs when label stack compression is disabled. When label stack compression is enabled, the result would be a set of compressed label stacks (composed of Adj-SIDs and node-SIDs) that provide *IP-like* ECMP forwarding behavior wherever possible.

For all computation results, we use an event-driven approach to provide updated results that are consistent with the current state of the network in a timely manner. However, we need to be careful to make sure that our computations do not become overwhelmed during those periods with large numbers of network events. Therefore the algorithm has the following properties:

- Very fast reaction for a single event (e.g., link failure)
- Fast-paced reaction for multiple IGP events that are temporally close, while computation and the ability to consume results are considered acceptable
- Delayed reaction when computation and ability to consume results are problematic

Furthermore, reaction to certain network events varies depending on whether label stack compression is enabled or not. For the following events, there is no immediate recomputation of SID-lists of candidate paths when compression is off:

- Change in TE-metric of links
- Link Down events where link is not traversed by candidate path
- Link Up event

When label stack compression is on, the above events are acted upon to see if the computation results are impacted.

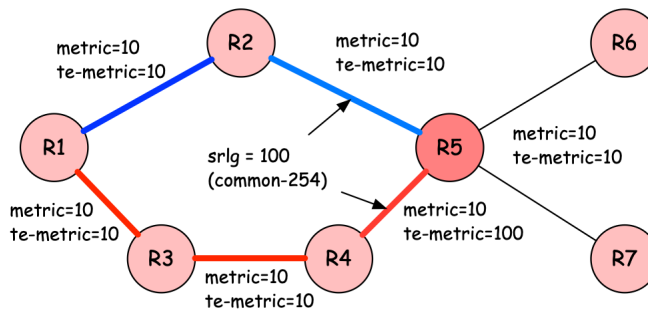


Figure 4.10

Simple Topology for SR CSPF Example

R1: CSPF

```
protocols {
  source-packet-routing {
    compute-options {
      optimize-timer 3600;
    }
    compute-profile blue-lsp {
      admin-group include-any blue;
    }
    compute-profile red-lsp {
      admin-group include-any red;
    }
  }
  source-routing-path sr-te-lsp-to-r6 {
    to 1.1.1.6;
    primary {
      1st-seg-to-r6 {
        compute red-lsp;
      }
    }
    secondary {
      2nd-seg-to-r6 {
        compute blue-lsp;
      }
    }
  }
}
```

This configuration results in the SR-TE LSPs being created as shown in Figure 4.11.

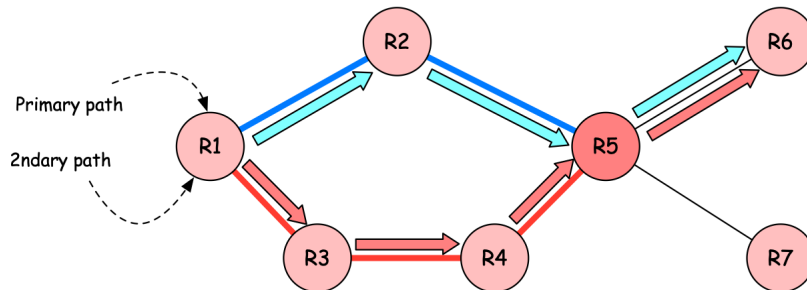


Figure 4.11

Resulting SR-TE LSPs

Verifying the SR-TE LSPs computed by the Distributed SR CSPF

```

user@R1# run show spring-traffic-engineering lsp detail
Name: sr-te-lsp-to-r6
Tunnel-source: Static configuration
To: 1.1.1.6
State: Up
  Path: 1st-seg-to-r6
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
  Compute Status:Enabled , Compute Result:success , Compute-Profile Name:red-lsp
  Total number of computed paths: 1
    Computed-path-index: 1
  BFD status: N/A BFD name: N/A
    computed segments count: 2
      computed segment : 1 (computed-node-segment):
        node segment label: 104
        router-id: 1.1.1.4
      computed segment : 2 (computed-node-segment):
        node segment label: 100
        router-id: 1.1.1.6
  Path: 2nd-seg-to-r6
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
  Compute Status:Enabled , Compute Result:success , Compute-Profile Name:blue-lsp
  Total number of computed paths: 1
    Computed-path-index: 1
  BFD status: N/A BFD name: N/A
    computed segments count: 1
      computed segment : 1 (computed-node-segment):
        node segment label: 100
        router-id: 1.1.1.6
  
```

As you can see in the output, Junos computes a path consisting of only the node SID for R4 to meet the constraints for the red path and also determines that the blue path is just the shortest path and thus only uses the node SID for R6, instead of computing a fully qualified path of adjacency SIDs. In contrast, if you add the `no-label-stack-compression` keyword you can see how a fully qualified SID list, comprised of adjacency SIDs, is computed.

Configuration example for R1

```
protocols {
  source-packet-routing {
    compute-profile blue-lsp {
      admin-group include-any blue;
      no-label-stack-compression;
    }
    compute-profile red-lsp {
      admin-group include-any red;
      no-label-stack-compression;
    }
  }
}
```

Verifying the SR-TE LSPs computed by the Distributed SR CSPF

```
user@R1# run show spring-traffic-engineering lsp detail
Name: sr-te-lsp-to-r6
Tunnel-source: Static configuration
To: 1.1.1.6
State: Up
  Path: 1st-seg-to-r6
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
  Compute Status:Enabled , Compute Result:success , Compute-Profile Name:red-lsp
  Total number of computed paths: 1
  Computed-path-index: 1
  BFD status: N/A BFD name: N/A
  computed segments count: 4
    computed segment : 1 (computed-adjacency-segment):
      label: 16
      source router-id: 1.1.1.1, destination router-id: 1.1.1.3
      source interface-address: 10.1.13.1, destination interface-address: 10.1.13.2
    computed segment : 2 (computed-adjacency-segment):
      label: 18
      source router-id: 1.1.1.3, destination router-id: 1.1.1.4
      source interface-address: 10.1.34.1, destination interface-address: 10.1.34.2
    computed segment : 3 (computed-adjacency-segment):
      label: 20
      source router-id: 1.1.1.4, destination router-id: 1.1.1.5
      source interface-address: 10.1.45.1, destination interface-address: 10.1.45.2
    computed segment : 4 (computed-adjacency-segment):
      label: 24
      source router-id: 1.1.1.5, destination router-id: 1.1.1.6
      source interface-address: 10.1.56.1, destination interface-address: 10.1.56.2
  Path: 2nd-seg-to-r6
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
  Compute Status:Enabled , Compute Result:success , Compute-Profile Name:blue-lsp
  Total number of computed paths: 1
  Computed-path-index: 1
```

```

BFD status: N/A BFD name: N/A
computed segments count: 3
  computed segment : 1 (computed-adjacency-segment):
    label: 19
    source router-id: 1.1.1.1, destination router-id: 1.1.1.2
    source interface-address: 10.1.12.1, destination interface-address: 10.1.12.2
  computed segment : 2 (computed-adjacency-segment):
    label: 21
    source router-id: 1.1.1.2, destination router-id: 1.1.1.5
    source interface-address: 10.1.25.1, destination interface-address: 10.1.25.2
  computed segment : 3 (computed-adjacency-segment):
    label: 24
    source router-id: 1.1.1.5, destination router-id: 1.1.1.6
    source interface-address: 10.1.56.1, destination interface-address: 10.1.56.2

```

Lastly, as discussed above, maximum SID depth continues to play an important role during dynamic CSPF. Like the previous PCEP example, a maximum SID depth can be set for each specific compute-profile using the `maximum-segment-list-depth <value>` key words.

Configuration example for R1

```

protocols {
  source-packet-routing {
    compute-profile blue-lsp {
      admin-group include-any blue;
      maximum-segment-list-depth 8;
    }
  }
}

```

A Centralized Controller or PCE for External CSPF

When discussing SR there is oftentimes an assumption that a controller or centralized PCE is present, so let's briefly explore how the Northstar TE Controller can be leveraged as an external path computation source for SR paths.

BGP Link-state for Topology Discovery

In order for the controller (PCE) to do any kind of path computation, it must be synchronized with the network “topology.” A network topology can take the form of a traffic engineering database (TED), much like what RSVP-TE uses for path computation, a link-state database (LSDB), or even more physical forms. The following example shows how using BGP-LS will convey a TED to Juniper's Northstar Controller.

NOTE Please refer to the *Visualization* section in Chapter 3 for a BGP-LS configuration example.

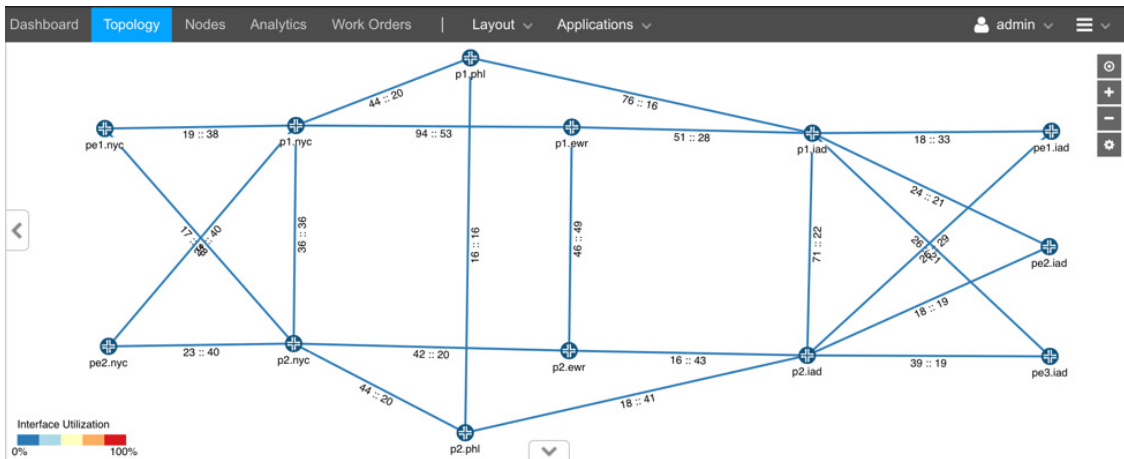
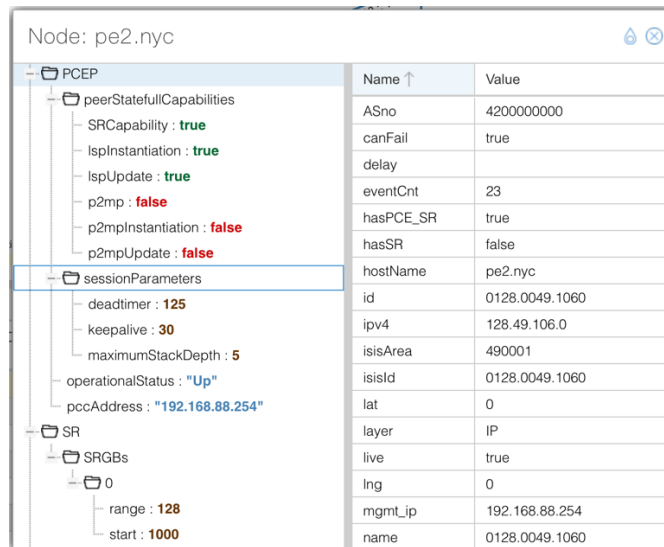


Figure 4.12 Graphical Display of Network Topology

PCEP for SR-TE LSP Creation and Control

The next relevant piece of information a controller requires is to be able to learn or to create a SR-TE LSP state. In next configuration example shows a Path Computation Element Protocol (PCEP) session between a Path Computation Client (PCC), or ingress router, and the controller. Figure 4.13 shows the PCEP session parameters.

```
protocols {
  source-routing {
    lsp-external-controller pccd;
  }
  pce NS1 {
    local-address 192.168.88.254;
    destination-ipv4-address 10.49.160.207;
    destination-port 4189;
    pce-type active stateful;
    lsp-provisioning;
    spring-capability;
  }
}
```



Name ↑	Value
ASno	4200000000
canFail	true
delay	
eventCnt	23
hasPCE_SR	true
hasSR	false
hostName	pe2.nyc
id	0128.0049.1060
ipv4	128.49.106.0
isisArea	490001
isisId	0128.0049.1060
lat	0
layer	IP
live	true
lng	0
mgmt_ip	192.168.88.254
name	0128.0049.1060

Figure 4.13 PCEP Session Parameters

Now let's revisit some of the SR-specific SID types and see how they 'look' on the controller and how to announce them.

Anycast SIDs are advertised by the IGP SR extensions during LSDb, by flooding, creating a second set of lo0.0 addresses and announcing a prefix-SID for them. The next example announces the anycast SID 123 from nodes p1.nyc and p2.nyc.

Announcing anycast SIDs from p1.nyc and p2.nyc

```

user@p1.nyc# run show configuration interfaces lo0.0
family inet {
    address 1.1.2.3/32;
}
user@p1.nyc# run show configuration policy-options policy-statement anycast-sid
term 0 {
    from {
        route-filter 1.1.2.3/32 exact;
    }
    then {
        prefix-segment index 123;
        accept;
    }
}
user@p1.nyc# run show configuration protocols isis
export anycast-sid;

```

Verifying on pe1.nyc

```

user@pe1.nyc> show isis database p2.nyc.00-00 extensive | match "1.1.2.3|123"
IP prefix: 1.1.2.3/32                      Metric:          0 Internal Up
IP prefix: 1.1.2.3/32, Internal, Metric: default 0, Up
IP extended prefix: 1.1.2.3/32 metric 0 up
Prefix SID, Flags: 0x00(R:0,N:0,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 123
IP address: 1.1.2.3

user@pe1.nyc> show isis database p1.nyc.00-00 extensive | match "1.1.2.3|123"
IP prefix: 1.1.2.3/32                      Metric:          0 Internal Up
IP prefix: 1.1.2.3/32, Internal, Metric: default 0, Up
IP extended prefix: 1.1.2.3/32 metric 0 up
Prefix SID, Flags: 0x00(R:0,N:0,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 123
IP address: 1.1.2.3

```

The anycast SID can be seen as a node property on the Northstar GUI, and later it's chosen as a strict or loose hop (more likely a loose hop) when creating the SR-TE LSP. Binding SIDs are added to a SR-TE LSP, as shown in Figure 4.14, and then advertised to Northstar via the PCE Protocol.

Node: p2.nyc

Name ↑	Value
ASno	4200000000
canFail	true
delay	
hasPCE_SR	false
hasSR	true
hostName	p2.nyc
id	0128.0049.1062
ipv4	128.49.106.2
isisArea	490001
isisId	0128.0049.1062
lat	0
layer	IP
live	true
lng	0

Node: p1.nyc

Name ↑	Value
ASno	4200000000
canFail	true
delay	
eventCnt	1
hasPCE_SR	true
hasSR	true
hostName	p1.nyc
id	0128.0049.1063
ipv4	1.1.2.3
isisArea	490001
isisId	0128.0049.1063
layer	IP
live	true
mgmt_ip	192.168.88.3

Figure 4.14

Node Properties Displaying Anycast SID

Creating a Core SR-TE LSP with a binding SID

```

user@p1.nyc# run show configuration protocols mpls label-range
static-label-range 1000000 1000100;

user@p1.nyc# run show configuration protocols source-packet-routing
lsp-external-controller pccd;
segment-list P1NYC-2-P1IAD {
    segment1 {
        label 1009;
        ip-address 128.49.106.9;
    }
}
source-routing-path P1NYC-2-P1IAD {
    to 128.49.106.9;
    binding-sid 1000001;
    metric 1;
    primary {
        P1NYC-2-P1IAD;
    }
}

```

Verifying the core LSP on P1.NYC

```

user@p1.nyc# run show spring-traffic-engineering lsp detail
Name: P1NYC-2-P1IAD
Tunnel-source: Static configuration
To: 128.49.106.9
State: Up
Telemetry statistics:
Sensor-name: ingress-P1NYC-2-P1IAD, Id: 3758096386
Sensor-name: transit-P1NYC-2-P1IAD, Id: 3758096387
Path: P1NYC-2-P1IAD
Outgoing interface: ge-0/0/4.0
Auto-translate status: Enabled Auto-translate result: Success
BFD status: N/A BFD name: N/A
SR-ERO hop count: 2
Hop 1 (Strict):
    NAI: IPv4 Adjacency ID, 0.0.0.0 -> 192.0.2.15
    SID type: 20-bit label, Value: 94
Hop 2 (Strict):
    NAI: IPv4 Adjacency ID, 0.0.0.0 -> 192.0.2.26
    SID type: 20-bit label, Value: 51

```

Total displayed LSPs: 1 (Up: 1, Down: 0)

Verifying the routing table on p1.nyc

```

user@p1.nyc# run show route protocol spring-te

inet.0: 49 destinations, 60 routes (46 active, 0 holddown, 3 hidden)

inet.3: 11 destinations, 15 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.9/32    *[SPRING-TE/8] 00:02:45, metric 1, metric2 20
>   to 192.0.2.21 via ge-0/0/5.0, Push 1009
>   to 192.0.2.13 via ge-0/0/1.0, Push 1009, Push 1008(top)

iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

```

```
mpls.0: 62 destinations, 62 routes (62 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1000001          *[SPRING-TE/8] 00:02:45, metric 1, metric2 20
> to 192.0.2.21 via ge-0/0/5.0, Swap 1009
  to 192.0.2.13 via ge-0/0/1.0, Swap 1009, Push 1008(top)
```

Verifying binding SIDs on the Northstar TE Controller

Binding SIDs can be verified using LSP properties or by adding a binding SID column to the Tunnels tab.

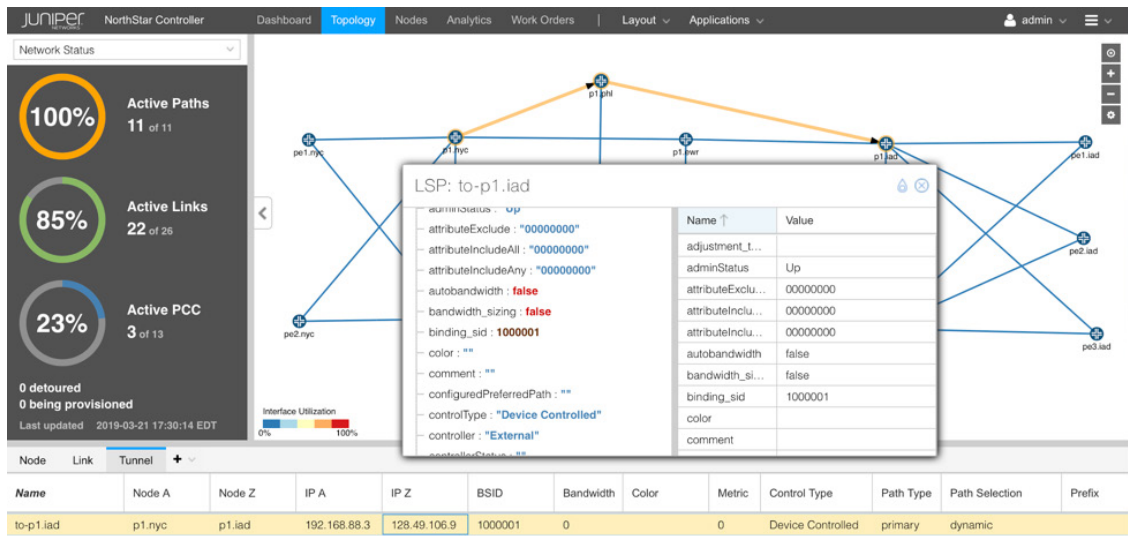


Figure 4.15 LSP Properties Displaying Binding SID

Now that you have explored various aspects of TE, several of the SR-specific SID types, and how basic information is synchronized with a controller, let’s revisit the sample network from previous chapters and bring an entire solution together!

End-to-End TE Solution

One of the key goals of transitioning the sample network to segment routing will be to replicate a form of “bandwidth optimization” (TE) into the core, Level 2 ISIS domain, where currently the Auto Bandwidth RSVP-TE LSPs provide a relatively granular per path bandwidth optimization. Because SR does not maintain the per LSP state, and thus per LSP statistics, a bandwidth optimization solution for a SR network requires that a controller acquire data from another source, such as streaming telemetry sources as described in Chapter 3, to end at a solution in another form that RSVP-TE would have.

First let's start by creating a mesh of SR-TE LSPs (see Table 4.1) between pe1.nyc and all the PEs in the IAD PoP. These LSPs will be created, ephemerally, using the PCE protocol such that the Northstar Controller has explicit control of each of their paths.

Table 4.1 SR-TE LSP Mesh

LSP Name	Ingress router	Egress router
pe1.nyc-pe1.iad	pe1.nyc	pe1.iad
pe1.nyc-pe2.iad	pe1.nyc	pe2.iad
pe1.nyc-pe3.iad	pe1.nyc	pe3.iad
pe1.iad-pe1.nyc	pe1.iad	pe1.nyc
pe2.iad-pe2.nyc	pe2.iad	pe1.nyc
pe3.iad-pe3.nyc	pe3.iad	pe1.nyc

To create a SR-TE LSP on the Northstar Controller use the Applications>Provision LSP drop-down option or the Add button on the Tunnel Tab. A pop-up window will appear, adding the attributes of the LSP, as shown in Figure 4.16. A key attribute when creating SR-TE LSPs is to provide them with some nominal 'Planned Bandwidth' value. This is to ensure that during periodic reoptimization, or a triggered reoptimization, link congestion awareness can be accounted for by Northstar's CSPF, as you will see later.

The screenshot shows the 'Provision LSP' window with the following configuration:

- Provisioning Method:** PCEP
- Name:** pe1.nyc-pe1.iad
- Node A:** pe1.nyc
- Node Z:** pe1.iad
- IP Z:** 128.49.106.11
- Provisioning Type:** SR
- Path Type:** primary
- Planned Bandwidth:** 10k
- Setup:** 7
- Hold:** 7
- Planned Metric:** (empty)
- Comment:** (empty)

Buttons at the bottom: Preview Path, Cancel, Submit.

Figure 4.16

Creating PCE Provisioned SR-TE LSPs Using PCEP

NOTE At the time of this writing, per SR policy ingress statistics were not available to the Northstar Controller. In the future, the static 10k bandwidth assigned to each SR-TE LSP can be replaced with a dynamic, real-time data plane statistics value so that a more granular bandwidth optimization is realizable.

The resulting SR-TE LSP mesh is shown in Figure 4.17. Here you can see what links are traversed by the mesh.

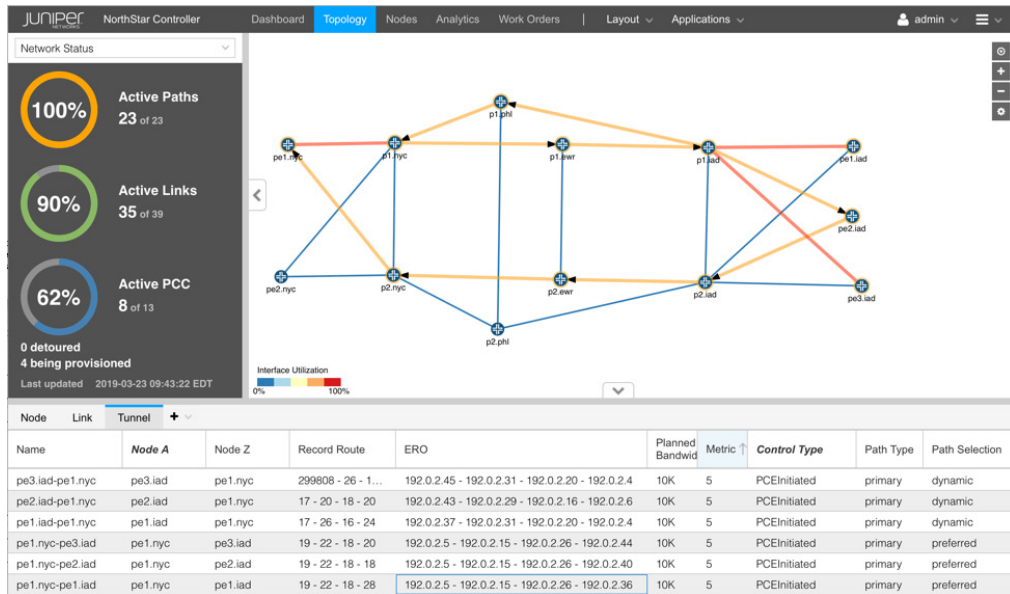


Figure 4.17 Sample Network SR Design

From the ingress PCC perspective, you can see that three SR-TE LSPs have been signaled via the Controller (PCE):

```
user@pe1.nyc> show path-computation-client lsp
```

Name	Status	PLSP-Id	LSP-Type	Controller	Path-Setup-Type
pe1.nyc-pe1.iad	Primary(Act)	10	ext-provided	NS1	spring-te
pe1.nyc-pe2.iad	Primary(Act)	11	ext-provided	NS1	spring-te
pe1.nyc-pe3.iad	Primary(Act)	12	ext-provided	NS1	spring-te

And that each has been installed in the inet.3 RIB of pe1.nyc:

```
user@pe1.nyc# run show route protocol spring-te table inet.3
```

```
inet.3: 8 destinations, 12 routes (8 active, 0 holddown, 0 hidden)
```

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

```
128.49.106.10/32 * [SPRING-TE/8] 00:06:23, metric 1, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 18, Push 18, Push 22(top)
128.49.106.11/32 [SPRING-TE/8] 1d 00:38:38, metric 1, metric2 0
```

```

> to 192.0.2.7 via ge-0/0/2.0, Push 28, Push 20, Push 18(top)
128.49.106.13/32 *[SPRING-TE/8] 00:05:30, metric 1, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 20, Push 20, Push 26(top)

```

You can verify the label stack for each SR-TE LSP via the Northstar Controller GUI, shown in the Figure 4.18, as shown in the Record Route and the ERO columns, and displaying the adjacency SIDs for the topology.

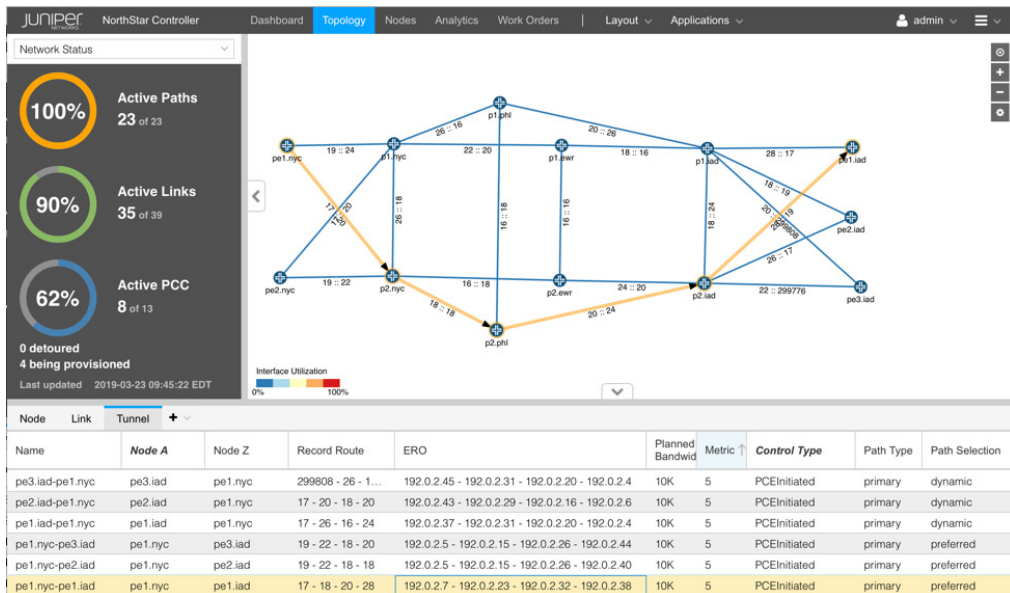


Figure 4.18 Verifying the Label Stack

As you can see from the Junos ingress router's SR-TE LSP detailed output, the label stack matches. It's worth noting that the first label (17, in the case of SR-TE LSP pe1.nyc-pe1.iad) is not actually imposed on the packet since the IP address from the PCEP NAI (node or adjacency identifier) field is used for the output interface selection. This is illustrated in the next output.

Detailed SR-TE LSP output:

```

user@pe1.nyc> show spring-traffic-engineering lsp detail
Name: pe1.nyc-pe1.iad
Tunnel-source: Path computation element protocol(PCEP)
To: 128.49.106.11
State: Up
Telemetry statistics:
Sensor-name: ingress-pe1.nyc-pe1.iad, Id: 3758096391
  Outgoing interface: NA
  Auto-translate status: Disabled Auto-translate result: N/A
  BFD status: N/A BFD name: N/A

```

```
SR-ERO hop count: 4
Hop 1 (Strict):
  NAI: IPv4 Adjacency ID, 192.0.2.6 -> 192.0.2.7
  SID type: 20-bit label, Value: 17
Hop 2 (Strict):
  NAI: IPv4 Adjacency ID, 192.0.2.22 -> 192.0.2.23
  SID type: 20-bit label, Value: 18
Hop 3 (Strict):
  NAI: IPv4 Adjacency ID, 192.0.2.33 -> 192.0.2.32
  SID type: 20-bit label, Value: 20
Hop 4 (Strict):
  NAI: IPv4 Adjacency ID, 192.0.2.39 -> 192.0.2.38
  SID type: 20-bit label, Value: 28
```

JUNOS inet.3 RIB entry for the SR-TE LSPs:

```
user@pe1.nyc# run show route protocol spring-te table inet.3
```

```
inet.3: 8 destinations, 12 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
128.49.106.10/32 * [SPRING-TE/8] 00:06:23, metric 1, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 18, Push 18, Push 22(top)
128.49.106.11/32 [SPRING-TE/8] 1d 00:38:38, metric 1, metric2 0
> to 192.0.2.7 via ge-0/0/2.0, Push 28, Push 20, Push 18(top)
128.49.106.13/32 * [SPRING-TE/8] 00:05:30, metric 1, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 20, Push 20, Push 26(top)
```

Our sample network is providing several services to attached CE routers. From ce1.nyc you can see the resulting label stack of the transport SR-TE LSPs between pe1.nyc and pe2.iad:

```
user@ce1.nyc> tracert 198.51.100.60
tracert to 198.51.100.60 (198.51.100.60), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1) 3.008 ms 1.918 ms 1.972 ms
 2 p1.nyc-ge-0-0-2.0 (192.0.2.5) 12.785 ms 9.685 ms 24.882 ms
    MPLS Label=22 CoS=0 TTL=1 S=0
    MPLS Label=18 CoS=0 TTL=1 S=0
    MPLS Label=18 CoS=0 TTL=1 S=1
 3 p1.ewr-ge-0-0-2.0 (192.0.2.15) 8.927 ms 14.411 ms 8.648 ms
    MPLS Label=18 CoS=0 TTL=1 S=0
    MPLS Label=18 CoS=0 TTL=2 S=1
 4 p1.iad-ge-0-0-6.0 (192.0.2.26) 9.039 ms 8.564 ms 10.233 ms
    MPLS Label=18 CoS=0 TTL=1 S=1
 5 pe2.iad-ge-0-0-1.0 (192.0.2.40) 7.857 ms 7.481 ms 17.570 ms
 6 ce1.iad-ge-0-0-7.0 (198.51.100.60) 8.767 ms 13.242 ms 15.533 ms
```

[illegible]

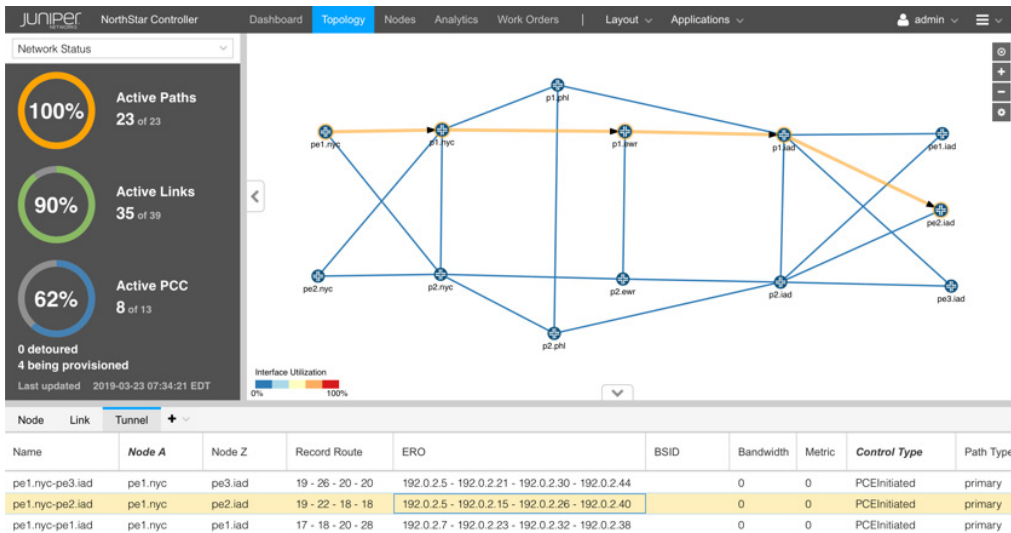


Figure 4.19 SR-TE LSP Path From pe1.nyc to pe2.iad

Now let's get back to how to provide a bandwidth optimization service for IS-IS Level 2 domain using the Northstar Controller. The original core network was built on RSVP-TE Auto Bandwidth LSPs. These dynamically adapt to increasing and decreasing traffic rates. Segment routing currently has no equivalent capabilities, primarily due to a lack of transit LSP state. We will enable an attribute on the Northstar Controller to react to interface congestion to trigger SR-TE LSP re-optimization based solely on ingress statistic collection. To enable this feature of the Northstar Controller go to Administration>Analytics and toggle the Reroute feature On as shown in Figure 4.20.

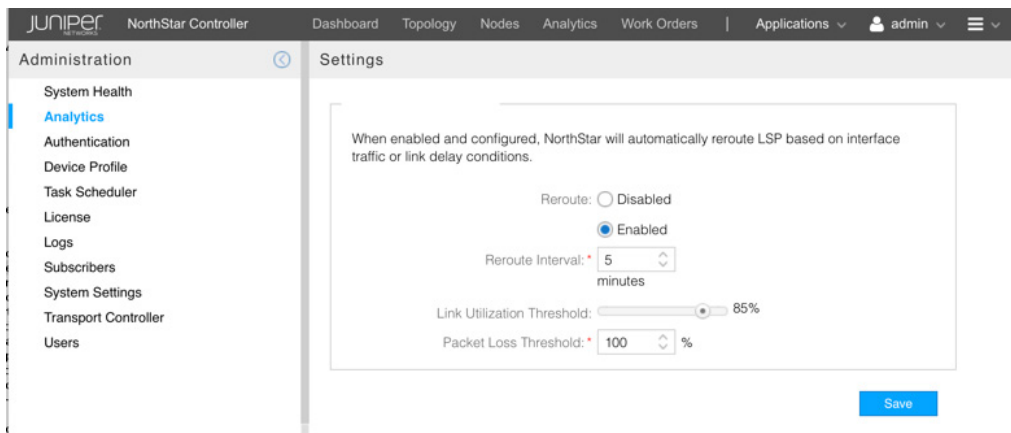


Figure 4.20 Enabling Interface Traffic-based Reroute

To ensure the topology also displays real-time interface statistics, use the drop down box on the left hand side of the GUI. Select Performance and enable Interface Utilization, as shown in Figure 4.21.

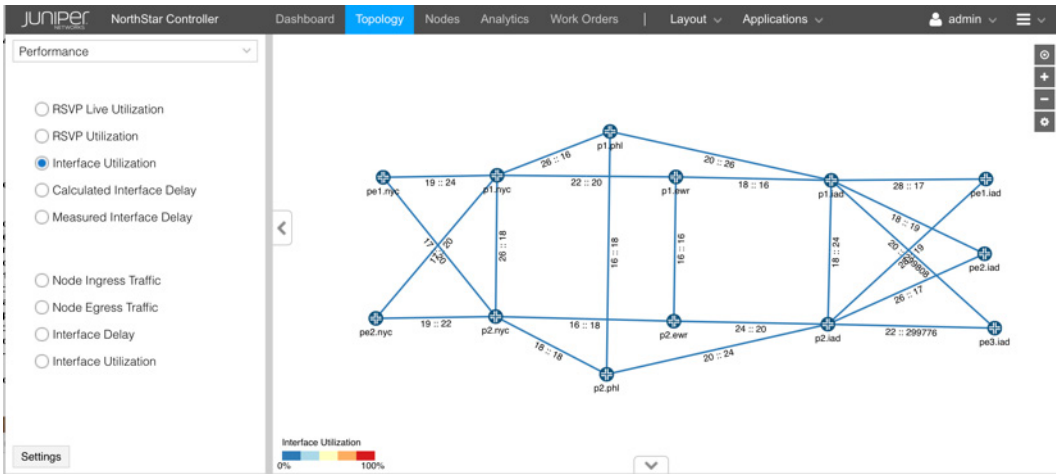


Figure 4.21 Enabling Interface Utilization-based Topology Display

To simulate traffic on the network, in order to illustrate the controller's ability to reroute SR-TE LSPs away from congested links, let's start some extended pings with large packet sizes between pe1.nyc and p1.iad:

```
user@pe1.nyc> traceroute 192.0.2.26
traceroute to 192.0.2.26 (192.0.2.26), 30 hops max, 40 byte packets
 1 p1.nyc-ge-0-0-2.0 (192.0.2.5) 3.569 ms 2.412 ms 3.261 ms
 2 p1.ewr-ge-0-0-2.0 (192.0.2.15) 6.769 ms 3.793 ms 3.152 ms
 3 p1.iad-ge-0-0-6.0 (192.0.2.26) 6.040 ms 5.814 ms 5.074 ms
```

```
user@pe1.nyc> ping rapid 192.0.2.26 count 10000000 size 500
```

```
PING 192.0.2.26 (192.0.2.26): 500 data bytes
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
[output truncated]
```

And likewise in the other direction...

```
user@p1.iad> traceroute 128.49.106.1
```

```
traceroute to 128.49.106.1 (128.49.106.1), 30 hops max, 40 byte packets
```

```
 1 p1.phl-ge-0-0-3.0 (192.0.2.31) 2.637 ms 2.077 ms 2.322 ms
 2 p1.nyc-ge-0-0-5.0 (192.0.2.20) 3.493 ms 5.646 ms 3.116 ms
 3 pe1.nyc-lo0.0 (128.49.106.1) 7.056 ms 5.973 ms 7.435 ms
```

```
user@p1.iad> ping 128.49.106.1 rapid count 10000000 size 1000
```

```
PING 128.49.106.1 (128.49.106.1): 1000 data bytes
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
[output truncated]
```

As you can see in Figure 4.22, Northstar detects the link congestion which will trigger a path reoptimization and reroute the SR-TE LSPs away from the congested link(s).

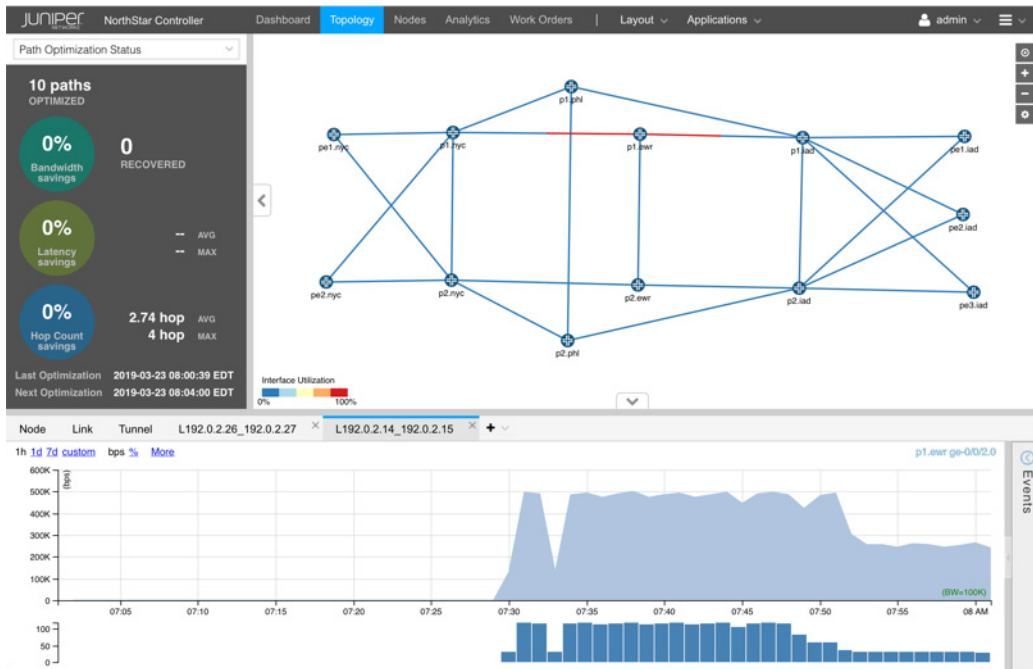


Figure 4.22 Link Congestion Display on Northstar GUI

By selecting Timeline in the left panel, you can see in Figure 4.23 that link congestion, based on the interface congestion threshold, has been detected and the LSPs that were traversing the congested link have been scheduled for rerouting.

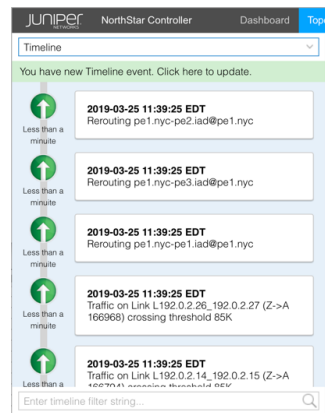


Figure 4.23 Time of Events for a Link Congestion Threshold Crossing

Going back to Northstar's Topology > Tunnel View, you can see the SR-TE LSP between pe1.nyc and pe2.iad in Figure 4.24. You can see that the LSP is now on a new path avoiding the congested link(s).

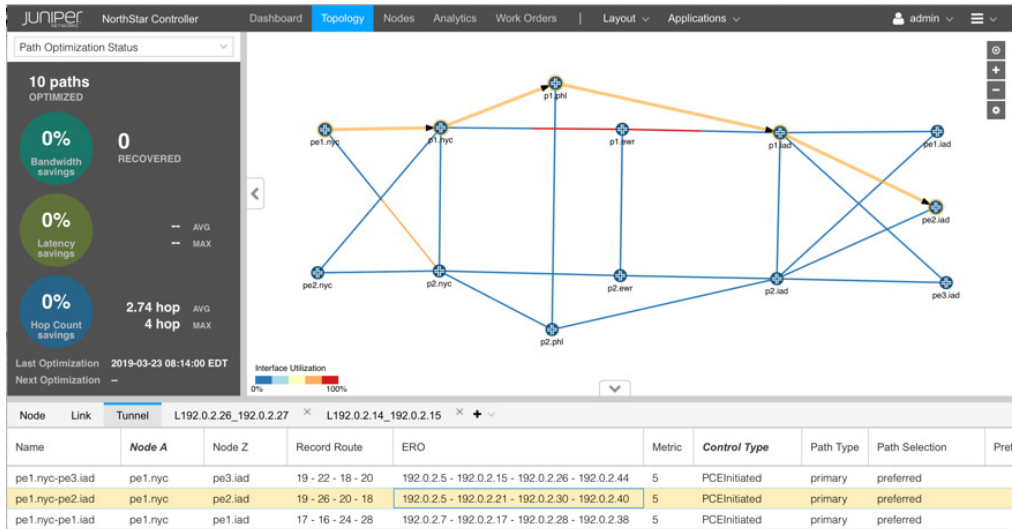


Figure 4.24 Updated SR-TE LSP Path After Congestion Detection

And the ingress router's SR-TE LSP has been updated as well:

```
user@pe1.nyc> show route table inet.3 protocol spring-te
```

```
inet.3: 8 destinations, 12 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
128.49.106.10/32  * [SPRING-TE/8] 00:00:07, metric 5, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 18, Push 20, Push 26(top)
128.49.106.11/32  [SPRING-TE/8] 00:00:07, metric 5, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 28, Push 18, Push 22(top)
128.49.106.13/32  * [SPRING-TE/8] 00:00:07, metric 5, metric2 0
> to 192.0.2.5 via ge-0/0/1.0, Push 20, Push 18, Push 22(top)
```

Traffic engineering is an indispensable function in most backbone wide area networks. A key objective of modern TE is the optimization of resource utilization. RSVP-TE has a long history and a large box of tools to leverage while SR needs to leverage newer tools, such as streaming telemetry and controllers, to achieve better resource utilization. This chapter provided a number of SR-specific options for creating explicit paths, various forms of distributed and centralized path computation, and finally, how our sample network can be transitioned to a SR TE. While the bandwidth optimization solution differs quite a bit from traditional RSVP-TE, it can provide a means of reacting to interface congestion to approximate resource optimization.

Chapter 5

Extensibility

You should now have a taste of everything a production segment-routed network is most likely to use. The set of tools in this chapter are a specialized lot. You will know if your network needs them.

SR over IP/UDP takes the stack of SIDs and zips them into an IP packet. This allows SR to be transported over an IP-only island. It also offers a non-MPLS data plane, and an alternative to SRv6.

FlexAlgo aims to democratize multi-topology routing through simplification. The SR concepts we have explored so far are retained, and now take place within a delineated topology. This opens up a number of interesting possibilities. The classic disjointed dual-plane service provider topology can be easily carved. 3GPP Release 15 espouses 5G network slicing. FlexAlgo can partition the SR-MPLS-enabled xHaul (mid or backhaul) and 5GC (5G core network).

Segment Routing Over IP/UDP

MPLS separates the underlay (transport) from the overlay (service) layers. Labels are commonly used to represent forwarding instructions for both layers. That said, the transport layer has always been capable of using non-MPLS tunneling technologies such as GRE, IPsec, or MPLS-over-IP/UDP tunnels, even as the service layer retains labels. SR over IP/UDP (shorted to SRoUDP) takes advantage of this.

A growing number of brownfield deployments are investigating tunneling a stack of SIDs over IP-only islands. SIDs remain relevant to SR-MPLS capable routers. Non-MPLS routers are oblivious to the SID stack and only need to perform IP forwarding. The IP islands – IPv4 or IPv6 – can remain as they are in perpetuity, allowing coexistence of SR-MPLS and IP-only networks.

SRoUDP works by segregating the network into SR-MPLS capable and incapable nodes. This is determined by the advertisement of, or lack thereof, a node SID. When an SR-MPLS capable router's next hop is IP-only, it encapsulates the SID stack into an IP/UDP packet. The IP packet is tunneled either to the next SR-MPLS capable router or the final destination based on the SID stack. Currently, Junos implements the latter.

Let's make the P routers in Newark and Philadelphia into IPv4-only routers. They should no longer advertise node SIDs, nor have LDP, or RSVP-TE enabled. This leaves the SR-MPLS capable P routers in New York and Washington connected over an IP expanse.

The New York PE routers will still impose one or more MPLS labels to reach the PEs in Washington. The local New York P routers that receive long-distance labeled traffic recognize that the destination is now reachable over IP next hops. They build a dynamic IP/UDP tunnel to the remote PE and encapsulate MPLS traffic within that. The Washington PE decapsulates the tunnel and forwards the payload.

This requires decapsulation configuration at the PE routers, and encapsulation configuration at the SR-MPLS P routers. The IP-only P routers require no reconfiguration.

Encapsulation configuration: New York and Washington P routers

```
user@p1.nyc> show configuration
routing-options {
  dynamic-tunnels {
    SR-O-UDP-ENCAP {
      source-address 128.49.106.3;
      udp;
      destination-networks {
        128.49.106.0/24;
      }
    }
  }
}
protocols {
  isis {
    source-packet-routing {
      udp-tunneling encapsulation;
    }
  }
}
...
```

Decapsulation configuration: New York and Washington PE routers

```
user@pe1.nyc> show configuration
interfaces {
  ge-0/0/1 {
    unit 0 {
      family inet {
        filter {
```

```

        input TRANSIT-TRAFFIC;
    }
}
}
ge-0/0/2 {
    unit 0 {
        family inet {
            filter {
                input TRANSIT-TRAFFIC;
            }
        }
    }
}
}
protocols {
    isis {
        source-packet-routing {
            udp-tunneling decapsulation;
        }
    }
}
firewall {
    family inet {
        filter TRANSIT-TRAFFIC {
            interface-specific;
            term MPLS-0-UDP {
                from {
                    protocol udp;
                    destination-port 6635;
                }
                then {
                    count MPLS-0-UDP;
                    decapsulate mpls-in-udp;
                }
            }
            term ALL {
                then {
                    count ALL;
                    accept;
                }
            }
        }
    }
}
...

```

The P routers are configured to create dynamic IP/UDP tunnels to any destination in the range that encompasses the addresses assigned to the routers' lo0. The `udp-tunneling encapsulation` knob allows IS-IS to include IP next hops during its route calculation. The PE routers are configured with a firewall filter that matches and decapsulates traffic on the well-known MPLS-over-UDP destination port. This filter is applied to all core-facing interfaces as the IP/UDP tunneled traffic could arrive over any of them.

After committing this change, something curious happens to the P routers' inet.3 table. Where before it only contained L-ISIS routes to area-local SR-MPLS routers, it additionally contains routes to remote P/PE routers. Let's take a look.

Control plane: p1.nyc adds routes to P/PE routers in D.C.

```
user@p1.nyc> show route table inet.3 active-path

inet.3: 9 destinations, 11 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.0/24    *[Tunnel/305] 00:01:36
                  Tunnel
128.49.106.0/32    *[L-ISIS/8] 1w1d 19:45:49, metric 10
                  > to 192.0.2.8 via ge-0/0/3.0, Push 0
                  > to 192.0.2.13 via ge-0/0/1.0, Push 1000
128.49.106.1/32    *[L-ISIS/8] 1w1d 19:45:49, metric 10
                  > to 192.0.2.4 via ge-0/0/2.0, Push 0
                  > to 192.0.2.13 via ge-0/0/1.0, Push 1001
128.49.106.2/32    *[L-ISIS/8] 1w1d 19:45:49, metric 10
                  > to 192.0.2.13 via ge-0/0/1.0
128.49.106.8/32    *[Tunnel/300] 00:01:36, metric 30, tag 1111
                  Tunnel Composite
128.49.106.9/32    *[Tunnel/300] 00:01:36, metric 20, tag 1111
                  Tunnel Composite
128.49.106.10/32   *[Tunnel/300] 00:01:36, metric 30, tag 1111
                  Tunnel Composite
128.49.106.11/32   *[Tunnel/300] 00:01:36, metric 30, tag 1111
                  Tunnel Composite
128.49.106.13/32   *[Tunnel/300] 00:01:36, metric 30, tag 1111
                  Tunnel Composite
```

Congruently, the mpls.0 table has an interesting label forwarding action associated with the remote routers' node SIDs. Instead of the typical *push*, *next*, and *continue* actions, you see the label being swapped for explicit-null and pushed into a dynamic IP/UDP tunnel. The explicit-null ensures that when pe1.iad receives the packet it performs an MPLS lookup on the inner packet.

Control plane: p1.nyc swaps inbound label 1010 (pe1.iad) with label 0 before tunneling

```
user@p1.nyc> show route label 1010 extensive

mpls.0: 52 destinations, 52 routes (52 active, 0 holddown, 0 hidden)
1010 (1 entry, 1 announced)
TSI:
KRT in-kernel 1010 /52 -> {indirect(1048589)}
    *L-ISIS Preference: 14
    ...
    Indirect next hops: 1
        Protocol next hop: 128.49.106.10 Metric: 30
        Label operation: Swap 0
        Load balance label: Label 0: None;
        Indirect next hop: 0xc92ad80 1048589 INH Session ID: 0x0
        Indirect path forwarding next hops: 1
            Next hop type: Tunnel Composite
            Next hop:
                128.49.106.10/32 Originating RIB: inet.3
                Metric: 30 Node path count: 1
                Forwarding nexthops: 1
```

```

Next hop type: Tunnel Composite
Tunnel type: UDP, nhid: 0, Reference-count: 4, tunnel id: 0
Destination address: 128.49.106.10, Source address: 128.49.106.3

```

Control plane: p1.nyc creates a dynamic tunnel to each SR-MPLS router in Washington

```

user@p1.nyc> show dynamic-tunnels database 128.49.106.10/32
*- Signal Tunnels #- PFE-down
Table: inet.3

```

```

Destination-network: 128.49.106.0/24
Tunnel to: 128.49.106.10/32
Reference count: 4
Next-hop type: UDP
Source address: 128.49.106.3
Next hop: tunnel-composite, 0xc8dd2b0, nhid 652
VPN Label: Swap 0 Reference count: 4
Ingress Route: 128.49.106.10/32, via metric 30
Traffic Statistics: Packets 10, Bytes 460
State: Up

```

The traceroute between the CEs is most transformed:

```

user@ce1.nyc> traceroute 198.51.100.54
traceroute to 198.51.100.54 (198.51.100.54), 30 hops max, 40 byte packets
 1 pe1.nyc-ge-0-0-10.1 (198.51.100.1) 3.027 ms 2.086 ms 2.248 ms
 2 pe1.iad-lo0.0 (128.49.106.11) 7.378 ms 6.197 ms 8.670 ms
 3 ce1.iad-ge-0-0-4.0 (198.51.100.54) 8.966 ms 9.399 ms 7.878 ms

```

Gone are the previously visible hops. Remember that the current behavior has p1.nyc establish a tunnel to the ultimate next hop; an alternative would be to establish a tunnel to the next closest SR-MPLS-capable router. With either approach, not all the transit routers are required to have an MPLS data plane. For many environments, that is a matter of fact. SRoUDP allows a SID stack to be transported over these IP islands.

FlexAlgo

Those who have dealt with multitopology routing (MTR) may shudder from its complexity. While not free of blemishes, MTR is a generally useful concept. The path less traveled may be favored over the shortest. Many operators maximize the IGP metric and optimize routing using the IGP traffic engineering metric. The TE metric may represent latency, which is arguably the metric that most affects typical end user experience.

MTR did not enjoy widespread acceptance. Practical implementations limited the number of topologies to an IPv4 and IPv6 topology, with unicast and multicast variants. More than anything, this failure in imagination is what led to its marginalization.

As the name implies, FlexAlgo aims to hand operators control over topology definition. All nodes and links participate in the default topology. Additional subset topologies are calculated by defining constraints and optimization objectives. To take a prototypical example, a red topology could include all links that are colored red, a familiar concept carried over from RSVP-TE; a supplementary blue topology could only include blue links. Finally, a purple topology could include those links that are both red and blue.

The first step is to define what constitutes a topology. With MTR, the operator would have to visit each router and configure each link's topological participation. This was an error-prone process, scaling poorly with growing networks. Discovering, validating, and enforcing membership in a topology required significant homegrown tooling.

FlexAlgo formalizes topology definition by advertising a Flexible Algorithm Definition (FAD) in the IGP itself. This attribute encodes which metric to calculate, which algorithm to use for the calculation, and most interestingly, link inclusion and exclusion based on extended administrative groups (EAG). Each FAD is numbered from 128-255; for operational convenience, it may be referred to nominally ('red' FAD, 'blue' FAD) but is ultimately encoded as a FAD number.

A FAD only need be configured on a subset of routers in the network. The remaining routers learn the FADs via the IGP. Each FAD also has an associated priority, with the highest numerical value being selected. The FADs must be identically configured across the subset of routers. In case of a conflict – one router advertising a 'red' FAD that includes links with red EAGs, and another advertising a 'red' FAD that accidentally includes links with blue EAGs – the highest priority FAD wins. As a tie breaker, the IGP system ID or router ID is used.

Once topology definitions have been learned, either via local configuration or IGP advertisement, routers can be configured to participate in those topologies. Again, this is not accomplished by configuring individual links, as in MTR. The FAD acts as a sieve for the default topology. The subset topologies are a result of each router pruning ineligible links, then optimizing for the metric type.

FlexAlgo can serve as an lightweight form of label compression. Rather than lengthy explicit paths, per-FAD node SIDs can provide loose-source routing with shallow label stacks.

An example will clarify. To keep things simple, let's configure an additional topology on pe1.nyc and p1.nyc. This new topology will optimize for the TE, instead of the IGP, metric. For now, it will not prune any links from the default topology. The resulting configuration numbers the FAD, specifies its optimization objective, and assigns it a priority:

```
user@pe1.nyc> show configuration
routing-options {
  flex-algorithm 128 {
```

```
metric-type te-metric;
priority 255;
...
```

```
user@p1.nyc> show configuration
routing-options {
  flex-algorithm 128 {
    metric-type te-metric;
    priority 254;
  }
}
```

This simply results in a new topology being defined without any routers participating in it yet. As a result, the FAD isn't even flooded by the IGP. The next step should be to have all New York routers participate in the topology – note that pe2.nyc and p2.nyc have learned about the FAD from IS-IS, not configuration:

```
user@pe1.nyc> show configuration
policy-options {
  policy-statement ISIS-EXPORT {
    ...
    term LOCAL-LOOPBACK {
      from {
        protocol direct;
        interface lo0.0;
        route-filter 128.49.106.1/32 exact;
      }
      then {
        tag 1111;
        prefix-segment {
          index 1;
          node-segment;
          add {
            128 {
              index 17;
              algorithm 128;
              node-segment;
            }
          }
        }
      }
    }
    accept;
  }
}
...
protocols {
  isis {
    source-packet-routing {
      flex-algorithm 128;
    }
    ...
    interface ge-0/0/1.0 {
      te-metric 1;
    }
    ...
    interface ge-0/0/2.0 {
      te-metric 100;
    }
  }
}
...

user@p1.nyc> show configuration
policy-options {
  policy-statement ISIS-EXPORT {
    ...
  }
}
```

```

term LOCAL-LOOPBACK {
  from {
    protocol direct;
    interface lo0.0;
    route-filter 128.49.106.3/32 exact;
  }
  then {
    tag 1111;
    prefix-segment {
      index 3;
      node-segment;
      add {
        128 {
          index 19;
          algorithm 128;
          node-segment;
        }
      }
    }
  }
  accept;
}
...
protocols {
  isis {
    source-packet-routing {
      flex-algorithm 128;
    }
    ...
    interface ge-0/0/1.0 {
      te-metric 100;
    }
    ...
    interface ge-0/0/2.0 {
      te-metric 1;
    }
  }
}
...

```

With two routers participating in the topology they are describing, multiple sub-TLVs now augment pe1.nyc and p1.nyc's IS-IS LSPs.

Control plane: Viewing pe1.nyc's IS-IS LSP on pe2.nyc

```

user@pe2.nyc> show isis database pe1.nyc extensive
IS-IS level 1 link-state database:

pe1.nyc.00-00 Sequence: 0x6d5, Checksum: 0x7e64, Lifetime: 1184 secs
  IPV4 Index: 1
  IPV4 Index: 17, IPV6 Index: 61
  ...
  IP extended prefix: 128.49.106.1/32 metric 0 up
  22 bytes of subtlvs
  Administrative tag 1: 1111
  Node SID, Flags: 0x70(R:0,N:1,P:1,E:1,V:0,L:0), Algo: SPF(0), Value: 1
  Node SID, Flags: 0x50(R:0,N:1,P:0,E:1,V:0,L:0), Algo: Unknown(128), Value: 17
  ...
  Router Capability: Router ID 128.49.106.1, Flags: 0x00
  SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 128, SID-Label: 1000
  SPRING Algorithm - Algo: 0
  SPRING Algorithm - Algo: 128
  Router Capability: Router ID 128.49.106.1, Flags: 0x01
  Flex Algo: 128, Len: 4, Metric: 1, Calc: 0, Prio: 255
  ...

```

With two routers participating in the topology they are describing, multiple sub-TLVs now augment pe1.nyc and p1.nyc's IS-IS LSPs. In addition to a new router capability that advertises the FAD itself, there is a new per-topology prefix SID being advertised.

This brings up another advantage of FlexAlgo when used to create dual or even multi-planar topologies. Often, for the same service route, BGP's protocol next hop (PNH) is modified via policy so it is different for each plane. RSVP-TE LSPs are spun to each PNH. The LSPs are routed differently, accomplishing dual-plane behavior. Other than the soft-state that RSVP-TE imposes, this additionally requires multiple IPv4 or IPv6 addresses to be allocated to each PE's loopback. Each address represents reachability to that PE, via a specific plane.

In contrast, FlexAlgo can assign a node SID per-topology. Each node SID remains associated with the same loopback IPv4 or IPv6 address. Service route PNH resolution can then be contained within different routing tables, each corresponding to a different topology. In this way, a single IP address can remain the PNH. Services routes can use different node SIDs to resolve the next hop, based on a combination of extended BGP color communities and their association with a FlexAlgo topology.

To demonstrate a simple dual-plane topology, let's have pe1.nyc prefer p1.nyc to reach pe2.nyc; in turn, pe2.nyc will prefer p2.nyc to reach pe1.nyc. Currently the PEs are reachable over equal-cost routes via the P routers. The earlier config snippet already set this up by using unequal TE metrics, which the new topology optimizes for. Go ahead and add per-topology node SID configuration to pe2.nyc and p2.nyc.

Control plane: pe1.nyc's new topology is a subset of the default topology

```
user@pe1.nyc> show route 128.49.106.0/32

inet.3: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.0/32    *[L-ISIS/14] 05:31:10, metric 20
                  to 192.0.2.5 via ge-0/0/1.0, Push 1000
                  > to 192.0.2.7 via ge-0/0/2.0, Push 1000

algo_128.inet.3: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

128.49.106.0/32    *[L-ISIS/14/5] 05:26:58, metric 11
                  > to 192.0.2.5 via ge-0/0/1.0, Push 1016
```

The new topology has a corresponding new routing information base (RIB). In that new topology, there are fewer available paths between pe1.nyc and pe2.nyc, and the metric between them is also different. Note the additional prefix SID (1000 + 16) advertised by pe2.nyc, associated with the same loopback address.

We began, and conclude, our journey with a simple example. Segment routing has reignited conversations about what it means to build flexible, efficient, and fungible networks.

Leap in.

Appendix

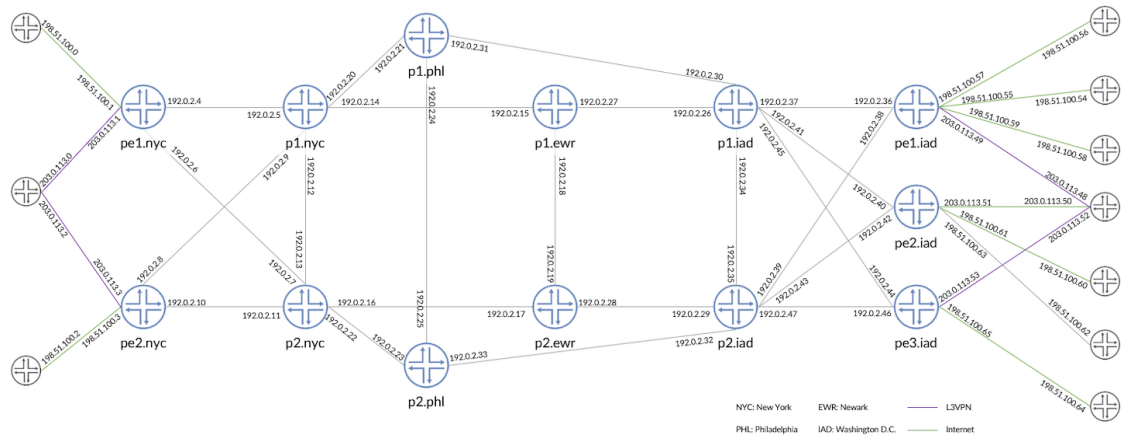


Figure A.1 Repeating Figure 1.1, This Book's Network Topology

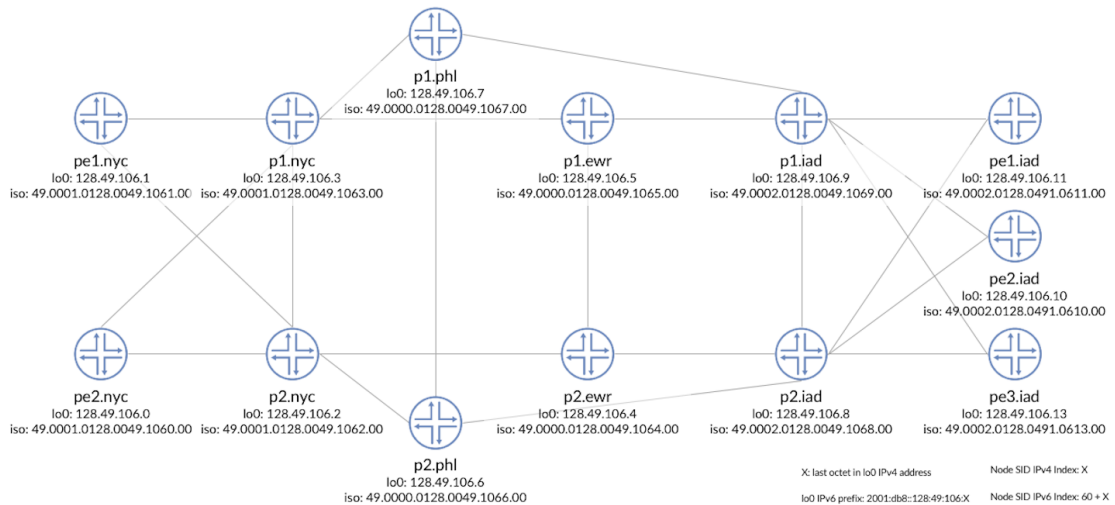


Figure A.2 Repeating Figure 2.4, IPv6, ISO Addressing and Additional Node SID Numbering

p1.ewr

```

## Last changed: 2019-03-23 04:55:54 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
    netconf {
      ssh;
    }
  }
}
RR {
  routing-options {
    router-id 128.49.106.5;
    autonomous-system 4200000000 asdot-notation;
  }
}
IGP {
  protocols {
    isis {
      level 1 disable;
      level 2 wide-metrics-only;
      interface all {
        point-to-point;
      }
      interface lo0.0 {
        passive;
        level 1 disable;
      }
      interface ge-0/0/0.0 {
        disable;
      }
      interface ge-0/0/1.0 {
        point-to-point;
        level 2 metric 999999;
      }
    }
  }
}
SR {
  protocols {
    isis {
      source-packet-routing {
        srgb start-label 1000 index-range 128;
        node-segment {
          ipv4-index 5;
          ipv6-index 65;
        }
      }
    }
  }
}

```

```

    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.5;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-transit-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
            }
        }
    }
    protocols {
        mpls {
            sensor-based-stats;
        }
    }
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
        }
    }
}

```

```

    }
    interface all {
        level 2 {
            post-convergence-lfa {
                node-protection;
            }
        }
    }
}
SR-TE-COLORED-ROUTES {
    protocols {
        bgp {
            group INTERNAL {
                family inet {
                    unicast {
                        extended-nexthop;
                        extended-nexthop-color;
                    }
                }
            }
            group INTERNAL_v6 {
                family inet6 {
                    unicast {
                        extended-nexthop-color;
                    }
                }
            }
        }
    }
}
}
}
}
}
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES RSVP-TE LOGGING SR IN-BAND PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];

```

p1.iad

```
## Last changed: 2019-03-22 14:51:27 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
  }
  netconf {
    ssh;
  }
}
}
```

```

}
RR {
  routing-options {
    router-id 128.49.106.9;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.9;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        cluster 128.49.106.9;
        allow 128.49.106.0/24;
        neighbor 128.49.106.3 {
          description p1.nyc;
        }
        neighbor 128.49.106.2 {
          description p2.nyc;
        }
      }
    }
  }
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {
          protocol direct;
          interface lo0.0;
          route-filter 128.49.106.0/24 prefix-length-range /32-/32;
        }
        then {
          tag 1111;
          accept;
        }
      }
      term LOCAL-LOOPBACK-V6 {
        from {
          family inet6;
          protocol direct;
          interface lo0.0;
          route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
        }
        then {
          tag 1111;
          accept;
        }
      }
      term L1-T0-L2 {
        from {
          protocol isis;
          level 1;
          tag 1111;

```

```

        }
        to level 2;
        then accept;
    }
    term L2-T0-L1 {
        from {
            protocol isis;
            level 2;
            tag 1111;
        }
        to level 1;
        then accept;
    }
    then reject;
}
}
protocols {
    isis {
        level 2 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/1.0 {
            point-to-point;
            level 2 metric 999999;
        }
        export ISIS-EXPORT;
    }
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 9;
                    ipv6-index 69;
                }
            }
        }
    }
}
LSP {
    protocols {
        mpls {
            label-switched-path p1.nyc:0 {
                to 128.49.106.3;
            }
            label-switched-path p2.nyc:0 {
                to 128.49.106.2;
            }
            label-history;
        }
    }
}
LSP-ATTRIBUTES {
    protocols {
        mpls {
            label-switched-path <*> {
                ldp-tunneling;
                record;
                oam {

```

```

        bfd-liveness-detection {
            minimum-interval 20000;
            minimum-receive-interval 20000;
            multiplier 3;
        }
        node-link-protection;
        adaptive;
    }
}
FA {
    protocols {
        isis {
            label-switched-path p1.nyc:0 {
                level 2 disable;
            }
            label-switched-path p2.nyc:0 {
                level 2 disable;
            }
        }
    }
}
ISIS-OVERLOAD {
    protocols {
        isis {
            overload;
        }
    }
}
SR-0-RSVP {
    protocols {
        isis {
            traffic-engineering {
                family inet-mpls {
                    shortcuts;
                }
            }
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:9;
                cluster 128.49.106.9;
                allow 2001:db8:0:0:128:49:106::/112;
            }
        }
    }
}
SR-0-RSVP_v6 {
    protocols {
        isis {
            traffic-engineering {
                family inet6-mpls {
                    shortcuts;
                }
            }
        }
    }
}
LSP-HIGH-METRIC {
    protocols {
        mpls {
            label-switched-path <*:0> {

```

```

        metric 16777215;
    }
}
}
BGP-LS {
    policy-options {
        policy-statement NLRI-T0-BGP-LS {
            term 1 {
                from family traffic-engineering;
                then accept;
            }
        }
        policy-statement TED-T0-NLRI {
            term LEVEL-1 {
                from {
                    protocol isis;
                    level 1;
                }
                then accept;
            }
            term LEVEL-2 {
                from {
                    protocol isis;
                    level 2;
                }
                then accept;
            }
        }
    }
    protocols {
        isis {
            traffic-engineering {
                inactive: igp-topology;
            }
        }
        mpls {
            traffic-engineering {
                database {
                    import {
                        igp-topology {
                            inactive: bgp-link-state;
                        }
                        policy TED-T0-NLRI;
                    }
                }
            }
        }
    }
    bgp {
        group NORTHSTAR {
            type internal;
            local-address 192.168.88.9;
            family traffic-engineering {
                unicast;
            }
            export NLRI-T0-BGP-LS;
            neighbor 192.168.88.99;
        }
    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {

```

```

        remote-address 192.168.88.100;
        remote-port 2001;
    }
    streaming-server ns-lsp {
        remote-address 192.168.88.100;
        remote-port 2002;
    }
    export-profile ns {
        local-address 128.49.106.9;
        local-port 2000;
        reporting-rate 30;
        format gpb;
        transport udp;
    }
    sensor ifd {
        server-name ns-ifd;
        export-name ns;
        resource /junos/system/linecard/interface/;
    }
    sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
    }
    sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
    }
    sensor sr-transit-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
    }
    }
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
            interface all {
                level 1 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
                level 2 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
            }
        }
    }
}
}

```

```

    }
  }
}
SR-TE-COLORED-ROUTES {
  protocols {
    bgp {
      group INTERNAL {
        family inet {
          unicast {
            extended-nexthop;
            extended-nexthop-color;
          }
        }
      }
      group INTERNAL_v6 {
        family inet6 {
          unicast {
            extended-nexthop-color;
          }
        }
      }
    }
  }
}
PCEP {
  protocols {
    mpls {
      lsp-external-controller pccd;
    }
    source-packet-routing {
      lsp-external-controller pccd;
      telemetry {
        statistics;
      }
    }
    pcep {
      pce NORTHSTAR {
        local-address 192.168.88.9;
        destination-ipv4-address 192.168.88.100;
        destination-port 4189;
        pce-type active stateful;
        lsp-provisioning;
        spring-capability;
      }
    }
  }
}
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES RSVP-TE LSP-ATTRIBUTES FA LOGGING SR SR-0-RSVP RR_v6 SR-0-RSVP_v6 LSP-HIGH-METRIC IN-
BAND BGP-LS TELEMETRY PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES PCEP ];

```

p1.nyc

```

## Last changed: 2019-03-23 04:54:48 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {

```

```

    ssl {
      port 32767;
      local-certificate local-cert;
      mutual-authentication {
        certificate-authority root_ca;
        client-certificate-request require-certificate-and-verify;
      }
      max-connections 30;
      inactive: skip-authentication;
    }
  }
}
netconf {
  ssh;
}
}
}
RR {
  routing-options {
    router-id 128.49.106.3;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.3;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        cluster 128.49.106.3;
        allow 128.49.106.0/24;
        neighbor 128.49.106.9 {
          description p1.iad;
        }
        neighbor 128.49.106.8 {
          description p2.iad;
        }
      }
    }
  }
}
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {
          protocol direct;
          interface lo0.0;
          route-filter 128.49.106.0/24 prefix-length-range /32-/32;
        }
        then {
          tag 1111;
          accept;
        }
      }
    }
    term LOCAL-LOOPBACK-V6 {
      from {
        family inet6;
      }
    }
  }
}

```

```

        protocol direct;
        interface lo0.0;
        route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
    }
    then {
        tag 1111;
        accept;
    }
}
term L1-T0-L2 {
    from {
        protocol isis;
        level 1;
        tag 1111;
    }
    to level 2;
    then accept;
}
term L2-T0-L1 {
    from {
        protocol isis;
        level 2;
        tag 1111;
    }
    to level 1;
    then accept;
}
    then reject;
}
}
protocols {
    isis {
        level 2 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/1.0 {
            point-to-point;
            level 2 metric 999999;
        }
        export ISIS-EXPORT;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 3;
                    ipv6-index 63;
                }
            }
            level 1 labeled-preference 8;
        }
    }
}
}
LSP-ATTRIBUTES {
    protocols {
        mpls {
            label-switched-path <*> {

```

```

        ldp-tunneling;
        record;
        oam {
            bfd-liveness-detection {
                minimum-interval 20000;
                minimum-receive-interval 20000;
                multiplier 3;
            }
        }
        node-link-protection;
        adaptive;
    }
}
}
}
LSP-IAD {
    protocols {
        mpls {
            label-switched-path p1.iad:0 {
                to 128.49.106.9;
            }
            label-switched-path p2.iad:0 {
                to 128.49.106.8;
            }
        }
    }
}
FA {
    protocols {
        isis {
            label-switched-path p1.iad:0 {
                level 2 disable;
            }
            label-switched-path p2.iad:0 {
                level 2 disable;
            }
        }
    }
}
SRMS {
    routing-options {
        source-packet-routing {
            mapping-server-entry srms.p1 {
                prefix-segment-range 128.49.106/24 {
                    start-prefix 128.49.106.10/32;
                    start-index 10;
                    size 4;
                }
            }
        }
    }
    protocols {
        isis {
            source-packet-routing mapping-server srms.p1;
        }
    }
}
SRMC {
    protocols {
        isis {
            source-packet-routing ldp-stitching;
        }
    }
}
LDP-SR-STITCHING {
    protocols {
        ldp {
            sr-mapping-client;
        }
    }
}

```

```

    }
}
LSP-ATTRIBUTES-AUTOBW {
    protocols {
        mpls {
            statistics {
                auto-bandwidth;
            }
            label-switched-path <*> {
                auto-bandwidth {
                    adjust-interval 300;
                    adjust-threshold 1;
                    adjust-threshold-absolute 100;
                    adjust-threshold-activate-bandwidth 1k;
                    maximum-bandwidth 1m;
                    adjust-threshold-overflow-limit 3;
                    adjust-threshold-underflow-limit 3;
                    resignal-minimum-bandwidth;
                }
            }
        }
    }
}
SR-0-RSVP {
    protocols {
        isis {
            traffic-engineering {
                family inet-mpls {
                    shortcuts;
                }
            }
        }
    }
}
LSP-HIGH-METRIC {
    protocols {
        mpls {
            label-switched-path <*:0> {
                metric 16777215;
            }
        }
    }
}
LSP-EWR {
    protocols {
        mpls {
            label-switched-path p1.ewr:0 {
                to 128.49.106.5;
            }
            label-switched-path p2.ewr:0 {
                to 128.49.106.4;
            }
        }
    }
}
LSP-PHL {
    protocols {
        mpls {
            label-switched-path p1.phl:0 {
                to 128.49.106.7;
            }
            label-switched-path p2.phl:0 {
                to 128.49.106.6;
            }
        }
    }
}
SR-RSVP-BDARK {
    interfaces {

```

```

        "<ge-0/0/[1-9]>" {
            unit <*> {
                bandwidth 100k;
            }
        }
    }
    routing-options {
        auto-bandwidth {
            template {
                SR-RSVP-COEXIST {
                    adjust-interval 30;
                    adjust-threshold 1;
                    statistic-collection-interval 10;
                }
            }
        }
    }
    protocols {
        isis {
            source-packet-routing {
                traffic-statistics {
                    statistics-granularity per-interface;
                    auto-bandwidth SR-RSVP-COEXIST;
                }
            }
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:3;
                cluster 128.49.106.3;
                allow 2001:db8:0:0:128:49:106::/112;
                neighbor 2001:db8::128:49:106:9 {
                    description p1.iad;
                }
                neighbor 2001:db8::128:49:106:8 {
                    description p2.iad;
                }
            }
        }
    }
}
SR-0-RSVP_v6 {
    protocols {
        isis {
            traffic-engineering {
                family inet6-mpls {
                    shortcuts;
                }
            }
        }
    }
}
PCEP {
    protocols {
        mpls {
            lsp-external-controller pccd;
        }
        source-packet-routing {
            lsp-external-controller pccd;
            telemetry {
                statistics;
            }
        }
    }
    pcep {

```

```

        pce NORTHSTAR {
            local-address 192.168.88.3;
            destination-ipv4-address 192.168.88.100;
            destination-port 4189;
            pce-type active stateful;
            lsp-provisioning;
            spring-capability;
        }
    }
}
BGP-LS {
    policy-options {
        policy-statement NLRI-T0-BGP-LS {
            term 1 {
                from family traffic-engineering;
                then accept;
            }
        }
        policy-statement TED-T0-NLRI {
            term LEVEL-1 {
                from {
                    protocol isis;
                    level 1;
                }
                then accept;
            }
            term LEVEL-2 {
                from {
                    protocol isis;
                    level 2;
                }
                then accept;
            }
        }
    }
    protocols {
        isis {
            traffic-engineering {
                inactive: igp-topology;
            }
        }
        mpls {
            traffic-engineering {
                database {
                    import {
                        igp-topology {
                            inactive: bgp-link-state;
                        }
                    }
                    policy TED-T0-NLRI;
                }
            }
        }
    }
    bgp {
        group NORTHSTAR {
            type internal;
            local-address 192.168.88.3;
            family traffic-engineering {
                unicast;
            }
            export NLRI-T0-BGP-LS;
            neighbor 192.168.88.99;
        }
    }
}
TELEMETRY {
    services {

```

```

analytics {
    streaming-server ns-ifd {
        remote-address 192.168.88.100;
        remote-port 2000;
    }
    streaming-server ns-ifl {
        remote-address 192.168.88.100;
        remote-port 2001;
    }
    streaming-server ns-lsp {
        remote-address 192.168.88.100;
        remote-port 2002;
    }
    export-profile ns {
        local-address 128.49.106.3;
        local-port 2000;
        reporting-rate 30;
        format gpb;
        transport udp;
    }
    sensor ifd {
        server-name ns-ifd;
        export-name ns;
        resource /junos/system/linecard/interface/;
    }
    sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
    }
    sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
    }
    sensor sr-transit-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
    }
}
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
            interface all {
                level 1 {
                    post-convergence-lfa {

```

```

    }
    level 2 {
        post-convergence-lfa {
            node-protection;
        }
    }
}
}
}
SR-TE-COLORED-ROUTES {
    protocols {
        bgp {
            group INTERNAL {
                family inet {
                    unicast {
                        extended-nexthop;
                        extended-nexthop-color;
                    }
                }
            }
            group INTERNAL_v6 {
                family inet6 {
                    unicast {
                        extended-nexthop-color;
                    }
                }
            }
        }
    }
}
}
}
}
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP INFRA_
INTERFACES MPLS RSVP-TE LSP-ATTRIBUTES FA SR LSP-ATTRIBUTES-AUTOBW LOGGING SR-O-RSVP LSP-HIGH-
METRIC LSP-EWR LSP-PHL SR-RSVP-BDARK RR_v6 SR-O-RSVP_v6 IN-BAND PCEP BGP-LS PKI TI-LFA-NODE-
PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];
```

p1.phl

Last changed: 2019-03-22 14:51:27 PDT

```
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
  }
  netconf {
    ssh;
  }
}
```

```

    }
  }
}
RR {
  routing-options {
    router-id 128.49.106.7;
    autonomous-system 4200000000 asdot-notation;
  }
}
IGP {
  protocols {
    isis {
      level 1 disable;
      level 2 wide-metrics-only;
      interface all {
        point-to-point;
      }
      interface lo0.0 {
        passive;
        level 1 disable;
      }
      interface ge-0/0/0.0 {
        disable;
      }
      interface ge-0/0/1.0 {
        point-to-point;
        level 2 metric 999999;
      }
    }
  }
}
SR {
  protocols {
    isis {
      source-packet-routing {
        srgb start-label 1000 index-range 128;
        node-segment {
          ipv4-index 7;
          ipv6-index 67;
        }
      }
    }
  }
}
}
TELEMETRY {
  services {
    analytics {
      streaming-server ns-ifd {
        remote-address 192.168.88.100;
        remote-port 2000;
      }
      streaming-server ns-ifl {
        remote-address 192.168.88.100;
        remote-port 2001;
      }
      streaming-server ns-lsp {
        remote-address 192.168.88.100;
        remote-port 2002;
      }
      export-profile ns {
        local-address 128.49.106.7;
        local-port 2000;
        reporting-rate 30;
        format gpb;
        transport udp;
      }
      sensor ifd {
        server-name ns-ifd;
        export-name ns;
      }
    }
  }
}

```

```

        resource /junos/system/linecard/interface/;
    }
    sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
    }
    sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
    }
    sensor sr-transit-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
    }
}
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
            interface all {
                level 2 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
            }
        }
    }
}
}
SR-TE-COLORED-ROUTES {
    protocols {
        bgp {
            group INTERNAL {
                family inet {
                    unicast {
                        extended-nexthop;
                        extended-nexthop-color;
                    }
                }
            }
            group INTERNAL_v6 {
                family inet6 {
                    unicast {
                        extended-nexthop-color;
                    }
                }
            }
        }
    }
}
}

```

```

    }
  }
}

apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES RSVP-TE LOGGING SR IN-BAND PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];

```

p2.ewr

```
## Last changed: 2019-03-22 14:51:27 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
    netconf {
      ssh;
    }
  }
}
RR {
  routing-options {
    router-id 128.49.106.4;
    autonomous-system 4200000000 asdot-notation;
  }
}
IGP {
  protocols {
    isis {
      level 1 disable;
      level 2 wide-metrics-only;
      interface all {
        point-to-point;
      }
      interface lo0.0 {
        passive;
        level 1 disable;
      }
      interface ge-0/0/0.0 {
        disable;
      }
      interface ge-0/0/1.0 {
        point-to-point;
        level 2 metric 999999;
      }
    }
  }
}
```

```

    }
  }
  SR {
    protocols {
      isis {
        source-packet-routing {
          srgb start-label 1000 index-range 128;
          node-segment {
            ipv4-index 4;
            ipv6-index 64;
          }
        }
      }
    }
  }
}
TELEMETRY {
  services {
    analytics {
      streaming-server ns-ifd {
        remote-address 192.168.88.100;
        remote-port 2000;
      }
      streaming-server ns-ifl {
        remote-address 192.168.88.100;
        remote-port 2001;
      }
      streaming-server ns-lsp {
        remote-address 192.168.88.100;
        remote-port 2002;
      }
      export-profile ns {
        local-address 128.49.106.4;
        local-port 2000;
        reporting-rate 30;
        format gpb;
        transport udp;
      }
      sensor ifd {
        server-name ns-ifd;
        export-name ns;
        resource /junos/system/linecard/interface/;
      }
      sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
      }
      sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
      }
      sensor sr-transit-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
      }
    }
  }
  protocols {
    mpls {
      sensor-based-stats;
    }
  }
}
}
OC_TELEMETRY {
  protocols {
    isis {
      source-packet-routing {

```

```
INTERFACES RSVP-TE LOGGING SR IN-BAND PKT TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ],
```

p2.iad

```
## Last changed: 2019-03-22 14:51:26 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```

    }
    max-connections 30;
    inactive: skip-authentication;
  }
}
}
netconf {
  ssh;
}
}
}
RR {
  routing-options {
    router-id 128.49.106.8;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.8;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        cluster 128.49.106.8;
        allow 128.49.106.0/24;
        neighbor 128.49.106.3 {
          description p1.nyc;
        }
        neighbor 128.49.106.2 {
          description p2.nyc;
        }
      }
    }
  }
}
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {
          protocol direct;
          interface lo0.0;
          route-filter 128.49.106.0/24 prefix-length-range /32-/32;
        }
        then {
          tag 1111;
          accept;
        }
      }
    }
    term LOCAL-LOOPBACK-V6 {
      from {
        family inet6;
        protocol direct;
        interface lo0.0;
        route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-

```

```

/128;
    }
    then {
        tag 1111;
        accept;
    }
}
term L1-T0-L2 {
    from {
        protocol isis;
        level 1;
        tag 1111;
    }
    to level 2;
    then accept;
}
term L2-T0-L1 {
    from {
        protocol isis;
        level 2;
        tag 1111;
    }
    to level 1;
    then accept;
}
then reject;
}
}
protocols {
    isis {
        level 2 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/1.0 {
            point-to-point;
            level 2 metric 999999;
        }
        export ISIS-EXPORT;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 8;
                    ipv6-index 68;
                }
            }
        }
    }
}
}
LSP {
    protocols {
        mpls {
            label-switched-path p1.nyc:0 {
                to 128.49.106.3;
            }
            label-switched-path p2.nyc:0 {
                to 128.49.106.2;
            }
        }
    }
}

```

```

        }
        label-history;
    }
}
LSP-ATTRIBUTES {
    protocols {
        mpls {
            label-switched-path <*> {
                ldp-tunneling;
                record;
                oam {
                    bfd-liveness-detection {
                        minimum-interval 20000;
                        minimum-receive-interval 20000;
                        multiplier 3;
                    }
                }
                node-link-protection;
                adaptive;
            }
        }
    }
}
FA {
    protocols {
        isis {
            label-switched-path p1.nyc:0 {
                level 2 disable;
            }
            label-switched-path p2.nyc:0 {
                level 2 disable;
            }
        }
    }
}
SR-0-RSVP {
    protocols {
        isis {
            traffic-engineering {
                family inet-mpls {
                    shortcuts;
                }
            }
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:8;
                cluster 128.49.106.8;
                allow 2001:db8:0:0:128:49:106::/112;
            }
        }
    }
}
SR-0-RSVP_v6 {
    protocols {
        isis {
            traffic-engineering {
                family inet6-mpls {
                    shortcuts;
                }
            }
        }
    }
}

```

```

    }
    LSP-HIGH-METRIC {
        protocols {
            mpls {
                label-switched-path <*:0> {
                    metric 16777215;
                }
            }
        }
    }
    BGP-LS {
        policy-options {
            policy-statement NLRI-T0-BGP-LS {
                term 1 {
                    from family traffic-engineering;
                    then accept;
                }
            }
            policy-statement TED-T0-NLRI {
                term LEVEL-1 {
                    from {
                        protocol isis;
                        level 1;
                    }
                    then accept;
                }
                term LEVEL-2 {
                    from {
                        protocol isis;
                        level 2;
                    }
                    then accept;
                }
            }
        }
        protocols {
            isis {
                traffic-engineering {
                    inactive: igp-topology;
                }
            }
            mpls {
                traffic-engineering {
                    database {
                        import {
                            igp-topology {
                                inactive: bgp-link-state;
                            }
                            policy TED-T0-NLRI;
                        }
                    }
                }
            }
            bgp {
                group NORTHSTAR {
                    type internal;
                    local-address 192.168.88.8;
                    family traffic-engineering {
                        unicast;
                    }
                    export NLRI-T0-BGP-LS;
                    neighbor 192.168.88.99;
                }
            }
        }
    }
    PCEP {
        protocols {
            mpls {

```

```

        lsp-external-controller NORTHSTAR;
    }
    source-packet-routing {
        lsp-external-controller NORTHSTAR;
        maximum-segment-list-depth 5;
        telemetry {
            statistics;
        }
    }
    pcep {
        pce NORTHSTAR {
            local-address 192.168.88.8;
            destination-ipv4-address 192.168.88.100;
            destination-port 4189;
            pce-type active stateful;
            lsp-provisioning;
            spring-capability;
            max-sid-depth 5;
        }
    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.8;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-transit-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
            }
        }
    }
}
protocols {
    mpls {
        sensor-based-stats;
    }
}

```

```

    }
  }
}
OC_TELEMETRY {
  protocols {
    isis {
      source-packet-routing {
        sensor-based-stats {
          per-interface-per-member-link ingress egress;
          per-sid ingress egress;
        }
      }
    }
  }
}
TI-LFA-NODE-PROTECT {
  protocols {
    isis {
      backup-spf-options {
        use-post-convergence-lfa;
      }
      interface all {
        level 1 {
          post-convergence-lfa {
            node-protection;
          }
        }
        level 2 {
          post-convergence-lfa {
            node-protection;
          }
        }
      }
    }
  }
}
SR-TE-COLORED-ROUTES {
  protocols {
    bgp {
      group INTERNAL {
        family inet {
          unicast {
            extended-nexthop;
            extended-nexthop-color;
          }
        }
      }
      group INTERNAL_v6 {
        family inet6 {
          unicast {
            extended-nexthop-color;
          }
        }
      }
    }
  }
}
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES LSP-ATTRIBUTES RSVP-TE FA LOGGING SR SR-0-RSVP RR_v6 SR-0-RSVP_v6 LSP-HIGH-METRIC BGP-
LS PCEP IN-BAND PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];

```

p2.nyc

Last changed: 2019-03-22 14:51:28 PDT

```

groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
    netconf {
      ssh;
    }
  }
}
RR {
  routing-options {
    router-id 128.49.106.2;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.2;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        cluster 128.49.106.2;
        allow 128.49.106.0/24;
        neighbor 128.49.106.9 {
          description p1.iad;
        }
        neighbor 128.49.106.8 {
          description p2.iad;
        }
      }
    }
  }
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {
          protocol direct;
          interface lo0.0;
        }
      }
    }
  }
}

```

```

        route-filter 128.49.106.0/24 prefix-length-range /32-/32;
    }
    then {
        tag 1111;
        accept;
    }
}
term LOCAL-LOOPBACK-V6 {
    from {
        family inet6;
        protocol direct;
        interface lo0.0;
        route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
    }
    then {
        tag 1111;
        accept;
    }
}
term L1-T0-L2 {
    from {
        protocol isis;
        level 1;
        tag 1111;
    }
    to level 2;
    then accept;
}
term L2-T0-L1 {
    from {
        protocol isis;
        level 2;
        tag 1111;
    }
    to level 1;
    then accept;
}
    then reject;
}
}
protocols {
    isis {
        level 2 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/1.0 {
            point-to-point;
            level 2 metric 999999;
        }
        export ISIS-EXPORT;
    }
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 2;
                    ipv6-index 62;
                }
            }
        }
    }
}

```

```

    }
    } level 1 labeled-preference 8;
  }
}
LSP-ATTRIBUTES {
  protocols {
    mpls {
      label-switched-path <*> {
        ldp-tunneling;
        record;
        oam {
          bfd-liveness-detection {
            minimum-interval 20000;
            minimum-receive-interval 20000;
            multiplier 3;
          }
        }
        node-link-protection;
        adaptive;
      }
    }
  }
}
LSP-IAD {
  protocols {
    mpls {
      label-switched-path p1.iad:0 {
        to 128.49.106.9;
      }
      label-switched-path p2.iad:0 {
        to 128.49.106.8;
      }
      label-history;
    }
  }
}
FA {
  protocols {
    isis {
      label-switched-path p1.iad:0 {
        level 2 disable;
      }
      label-switched-path p2.iad:0 {
        level 2 disable;
      }
    }
  }
}
SRMC {
  protocols {
    isis {
      source-packet-routing ldp-stitching;
    }
  }
}
SRMS {
  routing-options {
    source-packet-routing {
      mapping-server-entry srms.p2 {
        prefix-segment 128.49.106.10/32 index 10;
        prefix-segment 128.49.106.11/32 index 11;
        prefix-segment 128.49.106.13/32 index 13;
      }
    }
  }
  protocols {
    isis {

```

```

        source-packet-routing mapping-server srms.p2;
    }
}
LDP-SR-STITCHING {
    protocols {
        ldp {
            sr-mapping-client;
        }
    }
}
LSP-ATTRIBUTES-AUTOBW {
    protocols {
        mpls {
            statistics {
                auto-bandwidth;
            }
            label-switched-path <*> {
                auto-bandwidth {
                    adjust-interval 300;
                    adjust-threshold 1;
                    adjust-threshold-absolute 100;
                    adjust-threshold-activate-bandwidth 1k;
                    maximum-bandwidth 1m;
                    adjust-threshold-overflow-limit 3;
                    adjust-threshold-underflow-limit 3;
                    resignal-minimum-bandwidth;
                }
            }
        }
    }
}
SR-0-RSVP {
    protocols {
        isis {
            traffic-engineering {
                family inet-mpls {
                    shortcuts;
                }
            }
        }
    }
}
LSP-HIGH-METRIC {
    protocols {
        mpls {
            label-switched-path <*:0> {
                metric 16777215;
            }
        }
    }
}
LSP-EWR {
    protocols {
        mpls {
            label-switched-path p1.ewr:0 {
                to 128.49.106.5;
            }
            label-switched-path p2.ewr:0 {
                to 128.49.106.4;
            }
        }
    }
}
LSP-PHL {
    protocols {
        mpls {
            label-switched-path p1.phl:0 {
                to 128.49.106.7;
            }
        }
    }
}

```

```

        }
        label-switched-path p2.phl:0 {
            to 128.49.106.6;
        }
    }
}
SR-RSVP-BDARK {
    interfaces {
        "<ge-0/0/[1-9]>" {
            unit <*> {
                bandwidth 100k;
            }
        }
    }
    routing-options {
        auto-bandwidth {
            template {
                SR-RSVP-COEXIST {
                    adjust-interval 30;
                    adjust-threshold 1;
                    statistic-collection-interval 10;
                }
            }
        }
    }
    protocols {
        isis {
            source-packet-routing {
                traffic-statistics {
                    statistics-granularity per-interface;
                    auto-bandwidth SR-RSVP-COEXIST;
                }
            }
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:2;
                cluster 128.49.106.2;
                allow 2001:db8:0:0:128:49:106::/112;
                neighbor 2001:db8::128:49:106:9 {
                    description p1.iad;
                }
                neighbor 2001:db8::128:49:106:8 {
                    description p2.iad;
                }
            }
        }
    }
}
SR-0-RSVP_v6 {
    protocols {
        isis {
            traffic-engineering {
                family inet6-mpls {
                    shortcuts;
                }
            }
        }
    }
}
PCEP {
    protocols {
        mpls {

```

```

        lsp-external-controller NORTHSTAR;
    }
    source-packet-routing {
        lsp-external-controller NORTHSTAR;
        telemetry {
            statistics;
        }
    }
    pcep {
        pce NORTHSTAR {
            local-address 192.168.88.2;
            destination-ipv4-address 192.168.88.100;
            destination-port 4189;
            pce-type active stateful;
            lsp-provisioning;
            spring-capability;
        }
    }
}
BGP-LS {
    policy-options {
        policy-statement NLRI-T0-BGP-LS {
            term 1 {
                from family traffic-engineering;
                then accept;
            }
        }
        policy-statement TED-T0-NLRI {
            term LEVEL-1 {
                from {
                    protocol isis;
                    level 1;
                }
                then accept;
            }
            term LEVEL-2 {
                from {
                    protocol isis;
                    level 2;
                }
                then accept;
            }
        }
    }
    protocols {
        isis {
            traffic-engineering {
                inactive: igp-topology;
            }
        }
        mpls {
            traffic-engineering {
                database {
                    import {
                        igp-topology {
                            inactive: bgp-link-state;
                        }
                    }
                    policy TED-T0-NLRI;
                }
            }
        }
    }
    bgp {
        group NORTHSTAR {
            type internal;
            local-address 192.168.88.2;
            family traffic-engineering {
                unicast;
            }
        }
    }
}

```

```

    }
    export NLRI-T0-BGP-LS;
    neighbor 192.168.88.99;
}
}
}
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.2;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-transit-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
            }
        }
    }
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
TI-LFA-NODE-PROTECT {

```

p2.phl

p2.phl

```
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
  }
}
```

```

    }
    }
    netconf {
        ssh;
    }
}
RR {
    routing-options {
        router-id 128.49.106.6;
        autonomous-system 4200000000 asdot-notation;
    }
}
IGP {
    protocols {
        isis {
            level 1 disable;
            level 2 wide-metrics-only;
            interface all {
                point-to-point;
            }
            interface lo0.0 {
                passive;
                level 1 disable;
            }
            interface ge-0/0/0.0 {
                disable;
            }
            interface ge-0/0/1.0 {
                point-to-point;
                level 2 metric 999999;
            }
        }
    }
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 6;
                    ipv6-index 66;
                }
            }
        }
    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.6;
                local-port 2000;
                reporting-rate 30;
                format gpb;
            }
        }
    }
}

```

```

        transport udp;
    }
    sensor ifd {
        server-name ns-ifd;
        export-name ns;
        resource /junos/system/linecard/interface/;
    }
    sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
    }
    sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
    }
    sensor sr-transit-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/transit/usage/;
    }
}
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
            interface all {
                level 2 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
            }
        }
    }
}
}
SR-TE-COLORED-ROUTES {
    protocols {
        bgp {
            group INTERNAL {
                family inet {
                    unicast {
                        extended-nexthop;
                        extended-nexthop-color;
                    }
                }
            }
        }
    }
}
}

```

```

        group INTERNAL_v6 {
            family inet6 {
                unicast {
                    extended-nexthop-color;
                }
            }
        }
    }
}

apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES RSVP-TE LOGGING SR IN-BAND PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];

```

pe1.iad

```

## Last changed: 2019-03-22 14:41:30 PDT
groups {
    SERVICES {
        system {
            services {
                extension-service {
                    request-response {
                        grpc {
                            ssl {
                                port 32767;
                                local-certificate local-cert;
                                mutual-authentication {
                                    certificate-authority root_ca;
                                    client-certificate-request require-certificate-and-verify;
                                }
                            }
                        }
                    }
                    max-connections 30;
                    inactive: skip-authentication;
                }
            }
        }
        netconf {
            ssh;
        }
    }
}

PEERS_IPv4 {
    interfaces {
        ge-0/0/11 {
            description >ce1.iad:ge-0/0/4;
            unit 0 {
                description >198.51.100.54;
                family inet {
                    address 198.51.100.55/31;
                }
            }
        }
        ge-0/0/12 {
            description >ce1.iad:ge-0/0/5;
            unit 0 {
                description >198.51.100.56;
                family inet {
                    address 198.51.100.57/31;
                }
            }
        }
        ge-0/0/13 {

```

```

        description >ce1.iad:ge-0/0/6;
        unit 0 {
            description >198.51.100.58;
            family inet {
                address 198.51.100.59/31;
            }
        }
    }
}
RR {
    routing-options {
        router-id 128.49.106.11;
        autonomous-system 4200000000 asdot-notation;
    }
    protocols {
        bgp {
            group INTERNAL {
                type internal;
                local-address 128.49.106.11;
                family inet {
                    unicast;
                }
                family inet-vpn {
                    unicast;
                }
                family evpn {
                    signaling;
                }
                family route-target;
                neighbor 128.49.106.9 {
                    description p1.iad;
                }
                neighbor 128.49.106.8 {
                    description p2.iad;
                }
            }
        }
    }
}
IGP {
    policy-options {
        policy-statement ISIS-EXPORT {
            term OOB-MANAGEMENT-DENY {
                from interface [ em0.0 fxp0.0 ];
                then reject;
            }
            inactive: term PREFIX-SID-CONFLICT {
                from {
                    protocol direct;
                    interface lo0.0;
                    route-filter 128.49.106.9/32 exact;
                }
                then accept;
            }
            term LOCAL-LOOPBACK {
                from {
                    protocol direct;
                    interface lo0.0;
                    route-filter 128.49.106.0/24 prefix-length-range /32-/32;
                }
                then {
                    tag 1111;
                    accept;
                }
            }
            term LOCAL-LOOPBACK-V6 {
                from {
                    family inet6;
                }
            }
        }
    }
}

```

```

        protocol direct;
        interface lo0.0;
        route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
    }
    then {
        tag 1111;
        accept;
    }
    then reject;
}
}
protocols {
    isis {
        level 2 disable;
        level 1 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/11.0 {
            disable;
        }
        interface ge-0/0/12.0 {
            disable;
        }
        interface ge-0/0/13.0 {
            disable;
        }
        export ISIS-EXPORT;
        overload;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 11;
                    ipv6-index 71;
                }
            }
        }
    }
}
}
PEERS_IPv6 {
    interfaces {
        ge-0/0/11 {
            unit 0 {
                family inet6 {
                    address 2001:db8::198:51:100:55/127;
                }
            }
        }
        ge-0/0/12 {
            unit 0 {
                family inet6 {
                    address 2001:db8::198:51:100:57/127;
                }
            }
        }
    }
}
}

```

```

    ge-0/0/13 {
        unit 0 {
            family inet6 {
                address 2001:db8::198:51:100:59/127;
            }
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:11;
                export PEERS;
                neighbor 2001:db8::128:49:106:9 {
                    description p1.iad;
                }
                neighbor 2001:db8::128:49:106:8 {
                    description p2.iad;
                }
            }
        }
    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.11;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/tunnel/ingress/
usage/;
            }
        }
    }
}

```

```

    }
  }
  protocols {
    mpls {
      sensor-based-stats;
    }
  }
}
OC-TELEMETRY {
  protocols {
    isis {
      source-packet-routing {
        sensor-based-stats {
          per-interface-per-member-link ingress egress;
          per-sid ingress egress;
        }
      }
    }
  }
}
TI-LFA-NODE-PROTECT {
  protocols {
    isis {
      backup-spf-options {
        use-post-convergence-lfa;
      }
      interface all {
        level 1 {
          post-convergence-lfa {
            node-protection;
          }
        }
        level 2 {
          post-convergence-lfa {
            node-protection;
          }
        }
      }
    }
  }
}
BGP-EPE {
  protocols {
    bgp {
      egress-te-sid-stats;
      group PEERS {
        neighbor 198.51.100.56 {
          egress-te-node-segment {
            label 1044444;
          }
        }
      }
    }
  }
}
PREFIX-SID-CONFLICT {
  interfaces {
    lo0 {
      unit 0 {
        family inet {
          address 128.49.106.9/32;
        }
      }
    }
  }
  policy-options {
    policy-statement ISIS-EXPORT {
      term PREFIX-SID-CONFLICT {

```

```

    }
    }
    }
    SR-TE-COLORED-ROUTES {
        protocols {
            bgp {
                group INTERNAL {
                    family inet {
                        unicast {
                            extended-nexthop;
                            extended-nexthop-color;
                        }
                    }
                }
                group INTERNAL_v6 {
                    family inet6 {
                        unicast {
                            extended-nexthop-color;
                        }
                    }
                }
            }
        }
    }
    SR-TE-SBFD {
        protocols {
            bfd {
                sbfd {
                    local-discriminator 111 {
                        minimum-receive-interval 100;
                    }
                }
            }
        }
    }
}

apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 PE_CE_IPv4 PEERS_
IPv4 DNS RR IGP MPLS INFRA_INTERFACES SVC_VPNv4 SVC_IPv4 LOGGING SR PEERS_IPv6 PE_CE_IPv6 RR_v6 SVC_
IPv6 IN-BAND RSVP-TE PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES SR-TE-SBFD TELEMETRY ];
```

pe1.nyc

```
## Last changed: 2019-03-25 06:54:33 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
          }
        }
      }
    }
  }
}
max-connections 30;
inactive: skip-authentication;
```

```

    }
  }
  netconf {
    ssh;
  }
}
}
RR {
  routing-options {
    router-id 128.49.106.1;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.1;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        neighbor 128.49.106.3 {
          description p1.nyc;
        }
        neighbor 128.49.106.2 {
          description p2.nyc;
        }
      }
    }
  }
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {
          protocol direct;
          interface lo0.0;
          route-filter 128.49.106.0/24 prefix-length-range /32-/32;
        }
        then {
          tag 1111;
          accept;
        }
      }
      term LOCAL-LOOPBACK-V6 {
        from {
          family inet6;
          protocol direct;
          interface lo0.0;
          route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
        }
        then {
          tag 1111;
          accept;
        }
      }
    }
  }
}

```

```

        }
        then reject;
    }
}
protocols {
    isis {
        level 2 disable;
        level 1 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/10.1 {
            disable;
        }
        export ISIS-EXPORT;
        overload;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 1;
                    ipv6-index 61;
                }
                explicit-null;
            }
            level 1 labeled-preference 8;
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:1;
                export PEERS;
                neighbor 2001:db8::128:49:106:3 {
                    description p1.nyc;
                }
                neighbor 2001:db8::128:49:106:2 {
                    description p2.nyc;
                }
            }
        }
    }
}
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {

```

```

        remote-address 192.168.88.100;
        remote-port 2002;
    }
    export-profile ns {
        local-address 128.49.106.1;
        local-port 2000;
        reporting-rate 30;
        format gpb;
        transport udp;
    }
    sensor ifd {
        server-name ns-ifd;
        export-name ns;
        resource /junos/system/linecard/interface/;
    }
    sensor ifl {
        server-name ns-ifl;
        export-name ns;
        resource /junos/system/linecard/interface/logical/usage/;
    }
    sensor lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/label-switched-path/usage/;
    }
    sensor sr-lsp {
        server-name ns-lsp;
        export-name ns;
        resource /junos/services/segment-routing/traffic-engineering/ingress/usage/;
    }
}
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}
}
TI-LFA-NODE-PROTECT {
    protocols {
        isis {
            backup-spf-options {
                use-post-convergence-lfa;
            }
            interface all {
                level 1 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
                level 2 {
                    post-convergence-lfa {
                        node-protection;
                    }
                }
            }
        }
    }
}
}

```

```

}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 PE_CE_IPv4 DNS RR IGP MPLS INFRA_
INTERFACES SVC_VPNv4 SVC_IPv4 SR LOGGING PE_CE_IPv6 SVC_IPv6 RR_v6 IN-BAND RSVP-TE PKI TI-LFA-NODE-
PROTECT SR-TE-STATS SR-TE-COLORED-ROUTES TELEMETRY ];

```

pe2.iad

```
## Last changed: 2019-03-22 14:41:52 PDT
```

```

groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
  }
  netconf {
    ssh;
  }
}

```

```

}
PEERS_IPv4 {
    interfaces {
        ge-0/0/11 {
            description >ce1.iad:ge-0/0/7;
            unit 0 {
                description >198.51.100.60;
                family inet {
                    address 198.51.100.61/31;
                }
            }
        }
        ge-0/0/12 {
            description >ce1.iad:ge-0/0/8;
            unit 0 {
                description >198.51.100.62;
                family inet {
                    address 198.51.100.63/31;
                }
            }
        }
    }
}
RR {
    routing-options {
        router-id 128.49.106.10;
        autonomous-system 4200000000 asdot-notation;
    }
    protocols {
        bgp {
            group INTERNAL {
                type internal;
                local-address 128.49.106.10;
                family inet {
                    unicast;
                }
                family inet-vpn {
                    unicast;
                }
                family evpn {
                    signaling;
                }
                family route-target;
                neighbor 128.49.106.9 {
                    description p1.iad;
                }
                neighbor 128.49.106.8 {
                    description p2.iad;
                }
            }
        }
    }
}
IGP {
    policy-options {
        policy-statement ISIS-EXPORT {
            term OOB-MANAGEMENT-DENY {
                from interface [ em0.0 fxp0.0 ];
                then reject;
            }
            term LOCAL-LOOPBACK {
                from {
                    protocol direct;
                    interface lo0.0;
                    route-filter 128.49.106.0/24 prefix-length-range /32-/32;
                }
                then {
                    tag 1111;
                    accept;
                }
            }
        }
    }
}

```

```

    }
  }
  term LOCAL-LOOPBACK-V6 {
    from {
      family inet6;
      protocol direct;
      interface lo0.0;
      route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
    }
    then {
      tag 1111;
      accept;
    }
  }
  then reject;
}
}
protocols {
  isis {
    level 2 disable;
    level 1 wide-metrics-only;
    interface all {
      point-to-point;
    }
    interface lo0.0 {
      passive;
    }
    interface ge-0/0/0.0 {
      disable;
    }
    interface ge-0/0/11.0 {
      disable;
    }
    interface ge-0/0/12.0 {
      disable;
    }
    interface ge-0/0/13.0 {
      disable;
    }
    export ISIS-EXPORT;
    overload;
  }
}
}
SR {
  protocols {
    isis {
      source-packet-routing {
        srgb start-label 1000 index-range 128;
        node-segment {
          ipv4-index 10;
          ipv6-index 70;
        }
      }
    }
  }
}
}
PEERS_IPv6 {
  interfaces {
    ge-0/0/11 {
      unit 0 {
        family inet6 {
          address 2001:db8::198:51:100:61/127;
        }
      }
    }
    ge-0/0/12 {
      unit 0 {

```

```

        family inet6 {
            address 2001:db8::198:51:100:63/127;
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:10;
                export PEERS;
                neighbor 2001:db8::128:49:106:9 {
                    description p1.iad;
                }
                neighbor 2001:db8::128:49:106:8 {
                    description p2.iad;
                }
            }
        }
    }
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.10;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/tunnel/ingress/
usage/;
            }
        }
    }
}

```

```

        protocols {
            mpls {
                sensor-based-stats;
            }
        }
    }
    OC-TELEMETRY {
        protocols {
            isis {
                source-packet-routing {
                    sensor-based-stats {
                        per-interface-per-member-link ingress egress;
                        per-sid ingress egress;
                    }
                }
            }
        }
    }
    TI-LFA-NODE-PROTECT {
        protocols {
            isis {
                backup-spf-options {
                    use-post-convergence-lfa;
                }
                interface all {
                    level 1 {
                        post-convergence-lfa {
                            node-protection;
                        }
                    }
                    level 2 {
                        post-convergence-lfa {
                            node-protection;
                        }
                    }
                }
            }
        }
    }
    SR-TE-COLORED-ROUTES {
        protocols {
            bgp {
                group INTERNAL {
                    family inet {
                        unicast {
                            extended-nexthop;
                            extended-nexthop-color;
                        }
                    }
                }
                group INTERNAL_v6 {
                    family inet6 {
                        unicast {
                            extended-nexthop-color;
                        }
                    }
                }
            }
        }
    }
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 PE_CE_IPv4 PEERS_
IPv4 DNS RR IGP MPLS INFRA_INTERFACES SVC_VPNv4 SVC_IPv4 LOGGING_SR PEERS_IPv6 SVC_IPv6 PE_CE_
IPv6 RR_v6 IN-BAND RSVP-TE PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];

```

pe2.nyc

```

## Last changed: 2019-03-25 06:51:43 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
    netconf {
      ssh;
    }
  }
}
RR {
  routing-options {
    router-id 128.49.106.0;
    autonomous-system 4200000000 asdot-notation;
  }
  protocols {
    bgp {
      group INTERNAL {
        type internal;
        local-address 128.49.106.0;
        family inet {
          unicast;
        }
        family inet-vpn {
          unicast;
        }
        family evpn {
          signaling;
        }
        family route-target;
        neighbor 128.49.106.3 {
          description p1.nyc;
        }
        neighbor 128.49.106.2 {
          description p2.nyc;
        }
      }
    }
  }
}
IGP {
  policy-options {
    policy-statement ISIS-EXPORT {
      term OOB-MANAGEMENT-DENY {
        from interface [ em0.0 fxp0.0 ];
        then reject;
      }
      term LOCAL-LOOPBACK {
        from {

```

```

        protocol direct;
        interface lo0.0;
        route-filter 128.49.106.0/24 prefix-length-range /32-/32;
    }
    then {
        tag 1111;
        accept;
    }
}
term LOCAL-LOOPBACK-V6 {
    from {
        family inet6;
        protocol direct;
        interface lo0.0;
        route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;
    }
    then {
        tag 1111;
        accept;
    }
}
then reject;
}
}
protocols {
    isis {
        level 2 disable;
        level 1 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/10.1 {
            disable;
        }
        export ISIS-EXPORT;
        overload;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
                node-segment {
                    ipv4-index 0;
                    ipv6-index 60;
                }
                explicit-null;
            }
            level 1 labeled-preference 8;
        }
    }
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:0;
                export PEERS;
                neighbor 2001:db8::128:49:106:3 {

```

```

        description p1.nyc;
    }
    neighbor 2001:db8::128:49:106:2 {
        description p2.nyc;
    }
}
}
}
}
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.0;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/tunnel/ingress/
usage/;
            }
        }
    }
}
protocols {
    mpls {
        sensor-based-stats;
    }
}
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}

```

```

    }
  }
}
TI-LFA-NODE-PROTECT {
  protocols {
    isis {
      backup-spf-options {
        use-post-convergence-lfa;
      }
      interface all {
        level 1 {
          post-convergence-lfa {
            node-protection;
          }
        }
        level 2 {
          post-convergence-lfa {
            node-protection;
          }
        }
      }
    }
  }
}
}
SR-TE-STATS {
  protocols {
    source-packet-routing {
      telemetry {
        statistics;
      }
    }
  }
}
SR-TE {
  protocols {
    source-packet-routing {
      lsp-external-controller pccd;
      segment-list pe1.iad_v6_path0 {
        /* needs to be updated unless persistent adj-sids are used */
        hop1 label 24;
        hop2 {
          label 1061;
          label-type {
            node;
          }
        }
        hop3 {
          label 1071;
          label-type {
            node;
          }
        }
      }
      segment-list pe1.iad_v4_path0 {
        hop1 label 21;
        hop2 {
          label 1011;
          label-type {
            node;
          }
        }
      }
      segment-list pe1.iad_v4_path1_epe {
        hop1 label 21;
        hop2 {
          label 1011;
          label-type {
            node;
          }
        }
      }
    }
  }
}

```

```

        }
        hop3 label 1044444;
    }
    source-routing-path pe2.nyc:pe1.iad_v6 {
        to 2001:db8::128:49:106:11;
        color 7100;
        binding-sid 1000000;
        primary {
            pe1.iad_v6_path0;
        }
    }
    source-routing-path pe2.nyc:pe1.iad_v4 {
        to 128.49.106.11;
        color 7100;
        binding-sid 1000001;
        primary {
            pe1.iad_v4_path0;
        }
    }
}
}
}
SR-TE-COLORED-ROUTES {
    policy-options {
        policy-statement COLOR-ROUTES_v4 {
            term 1 {
                from {
                    protocol bgp;
                    route-filter 198.51.100.54/31 exact;
                }
                then {
                    community = HIGH-PRIORITY;
                    accept;
                }
            }
            term all {
                then accept;
            }
        }
        policy-statement COLOR-ROUTES_v6 {
            term 1 {
                from {
                    protocol bgp;
                    route-filter 2001:db8::198:51:100:54/127 exact;
                }
                then {
                    community = HIGH-PRIORITY;
                    accept;
                }
            }
            term all {
                then accept;
            }
        }
    }
    community HIGH-PRIORITY members color:0:7100;
}
protocols {
    bgp {
        group INTERNAL {
            import COLOR-ROUTES_v4;
            family inet {
                unicast {
                    extended-nexthop;
                    extended-nexthop-color;
                }
            }
        }
        group INTERNAL_v6 {
            import COLOR-ROUTES_v6;
        }
    }
}

```

```
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 PE_CE_IPv4 DNS RR IGP INFRA_
INTERFACES MPLS SVC_VPNv4 SVC_IPv4 SR LOGGING RR_v6 SVC_IPv6 PE_CE_IPv6 IN-BAND RSVP-TE PKI TI-LFA-
NODE-PROTECT SR-TE-STATS SR-TE SR-TE-COLORED-ROUTES SR-TE-SBFD TELEMETRY ];
```

pe3.iad

```
## Last changed: 2019-03-22 14:42:14 PDT
groups {
  SERVICES {
    system {
      services {
        extension-service {
          request-response {
            grpc {
              ssl {
                port 32767;
                local-certificate local-cert;
                mutual-authentication {
                  certificate-authority root_ca;
                  client-certificate-request require-certificate-and-verify;
                }
              }
            }
            max-connections 30;
            inactive: skip-authentication;
          }
        }
      }
    }
  }
  netconf {
    ssh;
  }
}
```

```

    }
}
PEERS_IPv4 {
    interfaces {
        ge-0/0/11 {
            description >ce1.iad:ge-0/0/9;
            unit 0 {
                description >198.51.100.64;
                family inet {
                    address 198.51.100.65/31;
                }
            }
        }
    }
}
RR {
    routing-options {
        router-id 128.49.106.13;
        autonomous-system 4200000000 asdot-notation;
    }
    protocols {
        bgp {
            group INTERNAL {
                type internal;
                local-address 128.49.106.13;
                family inet {
                    unicast;
                }
                family inet-vpn {
                    unicast;
                }
                family evpn {
                    signaling;
                }
                family route-target;
                neighbor 128.49.106.9 {
                    description p1.iad;
                }
                neighbor 128.49.106.8 {
                    description p2.iad;
                }
            }
        }
    }
}
IGP {
    policy-options {
        policy-statement ISIS-EXPORT {
            term OOB-MANAGEMENT-DENY {
                from interface [ em0.0 fxp0.0 ];
                then reject;
            }
            term LOCAL-LOOPBACK {
                from {
                    protocol direct;
                    interface lo0.0;
                    route-filter 128.49.106.0/24 prefix-length-range /32-/32;
                }
                then {
                    tag 1111;
                    accept;
                }
            }
            term LOCAL-LOOPBACK-V6 {
                from {
                    family inet6;
                    protocol direct;
                    interface lo0.0;
                    route-filter 2001:db8:0000:0000:128:49:106::/112 prefix-length-range /128-
/128;

```

```

        }
        then {
            tag 1111;
            accept;
        }
    }
    then reject;
}
}
protocols {
    isis {
        level 2 disable;
        level 1 wide-metrics-only;
        interface all {
            point-to-point;
        }
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/0.0 {
            disable;
        }
        interface ge-0/0/11.0 {
            disable;
        }
        interface ge-0/0/12.0 {
            disable;
        }
        interface ge-0/0/13.0 {
            disable;
        }
        export ISIS-EXPORT;
        overload;
    }
}
}
SR {
    protocols {
        isis {
            source-packet-routing {
                srgb start-label 1000 index-range 128;
            }
            node-segment {
                ipv4-index 13;
                ipv6-index 73;
            }
        }
    }
}
}
PEERS_IPv6 {
    interfaces {
        ge-0/0/11 {
            unit 0 {
                family inet6 {
                    address 2001:db8::198:51:100:65/127;
                }
            }
        }
    }
}
}
RR_v6 {
    protocols {
        bgp {
            group INTERNAL_v6 {
                type internal;
                local-address 2001:db8::128:49:106:13;
                export PEERS;
                neighbor 2001:db8::128:49:106:9 {
                    description p1.iad;
                }
            }
        }
    }
}
}

```

```

        }
        neighbor 2001:db8::128:49:106:8 {
            description p2.iad;
        }
    }
}
}
TELEMETRY {
    services {
        analytics {
            streaming-server ns-ifd {
                remote-address 192.168.88.100;
                remote-port 2000;
            }
            streaming-server ns-ifl {
                remote-address 192.168.88.100;
                remote-port 2001;
            }
            streaming-server ns-lsp {
                remote-address 192.168.88.100;
                remote-port 2002;
            }
            export-profile ns {
                local-address 128.49.106.13;
                local-port 2000;
                reporting-rate 30;
                format gpb;
                transport udp;
            }
            sensor ifd {
                server-name ns-ifd;
                export-name ns;
                resource /junos/system/linecard/interface/;
            }
            sensor ifl {
                server-name ns-ifl;
                export-name ns;
                resource /junos/system/linecard/interface/logical/usage/;
            }
            sensor lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/label-switched-path/usage/;
            }
            sensor sr-lsp {
                server-name ns-lsp;
                export-name ns;
                resource /junos/services/segment-routing/traffic-engineering/tunnel/ingress/
usage/;
            }
        }
    }
    protocols {
        mpls {
            sensor-based-stats;
        }
    }
}
OC_TELEMETRY {
    protocols {
        isis {
            source-packet-routing {
                sensor-based-stats {
                    per-interface-per-member-link ingress egress;
                    per-sid ingress egress;
                }
            }
        }
    }
}

```

```
}
apply-groups [ global member0 AAA SERVICES OAM CHASSIS FIB INFRA_IPv4 PE_CE_IPv4 PEERS_
IPv4 DNS RR IGP MPLS INFRA_INTERFACES SVC VPNv4 SVC_IPv4 LOGGING SR PEERS_IPv6 PE_CE_IPv6 SVC_
IPv6 RR v6 IN-BAND RSVP-TE PKI TI-LFA-NODE-PROTECT SR-TE-COLORED-ROUTES TELEMETRY ];
```