JUNIPER NETWORKS | Engineering Simplicity

DAY ONE: CONFIGURING SEGMENT ROUTING WITH JUNOS

Follow this book's lab to configure the basics of segment routing and then enable advanced traffic protection.

By Julian Lucek and Krzysztof Szarkowicz

DAY ONE: CONFIGURING SEGMENT ROUTING WITH JUNOS

Over the past few years there's been a lot of buzz in the industry about Segment Routing, and in response Juniper has developed a very comprehensive feature set in the Junos® OS and the NorthStar Controller. This book explains how to configure and understand the operation of Segment Routing, or SPRING, in the Junos OS.

In *Day One: Configuring Segment Routing with Junos*, two Senior Engineers distill their professional field experience and knowledge into a lab book well worth following. You'll learn the different types of segments and how to fine-tune them in various topologies: the first half of the book configures and examines the components that make SPRING so attractive, and the second half of the book instructs you on how to configure traffic engineering into the network you built in the first half.

"There has been a lot of interest in Segment Routing from hyperscalers, service providers, and large enterprises alike. This book takes you all the way from the fundamental principles of Segment Routing to advanced topics such as TI-LFA in an easy to follow, clearly explained format. SREs, network designers, and network architects will all make it their go-to reference."

Bikash Koley, EVP & CTO, Juniper Networks

"Configuring Segment Routing with Junos is the ideal book to understand and to implement Segment Routing technology in an IP/MPLS network. It describes the basic concepts of the technology as well as some implementation use cases like traffic protection with TI-LFA. The book provides several examples of configuration and operational commands which will definitely help you during your implementation. That is worth reading!"

Stephane Litkowski, Network Architect and Orange Expert, Orange Business Services

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

- Understand the key principles behind Segment Routing.
- Configure Segment Routing on routers running the Junos OS.
- Verify Segment Routing operation on Junos devices.
- Configure Segment Routing to LDP interworking and understand the operating principles.
- Understand and configure TI-LFA operation to support traffic protection.

ISBN 978-1-941441-73-2

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

JUNIPER NETWORKS

Day One: Configuring Segment Routing with Junos®

by Julian Lucek and Krzysztof Szarkowicz

Chapter 1: Principles of Segment Routing 9

Chapter 2: Adjacency Segments 14

Chapter 3: Node Segments and Anycast Segments 24

Chapter 4: Interworking Between Segment Routing and LDP 36

Chapter 5: Topology Independent Loop Free Alternate (TI-LFA) 43

JUNIPER NETWORKS

Published by Juniper Networks Books
Author: Julian Lucek and Krzysztof Szarkowicz
Technical Reviewers: Shelesh Bansal, Chris Bowers, Shraddha Hedge
Editor in Chief: Patrick Ames
Copyeditor: Nancy Koerbel

ISBN: 978-1-941441-73-2 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-74-9 (ebook)

Version History: v1, July 2018
2 3 4 5 6 7 8 9 10

http://www.juniper.net/dayone

About the Authors

Julian Lucek is a Distinguished Systems Engineer at Juniper Networks, where he has been working with many operators on the design and evolution of their networks. Before joining Juniper Networks in 1999, he worked at BT for several years, at first in the Photonics Research Department and later in the data transport and routing area. During this time he gained a PhD in ultrahigh-speed optical transmission and processing from Cambridge University. He holds several patents in the area of communications technology. He is the co-author of *MPLS-Enabled Applications: Emerging Developments and New Technologies* with Ina Minei (Wiley, 2011).

Krzysztof Grzegorz Szarkowicz is a Product Line Manager at Juniper Networks, and has been working with the networking industry 20+ years, delivering projects at many operators world wide. Before joining Juniper Networks in 2007, he worked with Hewlett Packard Labs, Telia Research, Ericsson, and Cisco. During that time he gained valuable hands-on experience with various aspects of networking, culminating in gaining CCIE-SP and JNCIE-SP certifications. He is the co-author of *MPLS in the SDN Era* with Antonio Sánchez-Monge (O'Reilly Media, 2015).

Authors' Acknowledgements

Julian Lucek: Many thanks to Patrick Ames, Editor in Chief, for his expert advice and keeping me on track with this project.

Krzysztof Szarkowicz: I would like to thank (in alphabetical order) Shelesh Bansal, Chris Bowers, and Shraddha Hedge from the Juniper Networks engineering team for many technical discussions and reviewing the material. And, naturally, many thanks to the editors for managing the entire book publishing process.

Welcome to Day One

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

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

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

You can obtain publications from either series in multiple formats:

- Download a free PDF edition at http://www.juniper.net/dayone.
- Get the ebook edition for iPhones and iPads from the iTunes/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 Kindle Store. Search for Juniper Networks Books or the title of this book.
- Purchase the paper edition at either Vervante Corporation (www.vervante.com) for between $15-$40, depending on page length.
- Note that most mobile devices can also view PDF files.

What You Need to Know Before Reading This Book

Before reading this book, you need to be familiar with the basic administrative functions of the Junos operating system, including the ability to work with operational commands and to read, understand, and change Junos configurations. There are several books in the *Day One Fundamentals Series* on learning the Junos OS, at http://www.juniper.net/dayone.

This book makes a few assumptions about you, the reader:

- You are familiar with and versed in using the Junos CLI for router configuration.
- You have a basic understanding of ISIS routing and MPLS.
- You can build out the lab topologies used in this book without detailed setup instructions.

What You Will Learn by Reading This Book

After reading this book you will be able to:

- Understand the key principles behind Segment Routing.
- Configure Segment Routing on routers running the Junos OS.
- Verify Segment Routing operation on Junos devices.
- Configure Segment Routing to LDP interworking and understand the operating principles.
- Understand and configure TI-LFA operation to support traffic protection.

About This Book

Welcome to this *Day One* book on Segment Routing (SR), also known as SPRING. The aim of this book is to provide a grounding in the key principles of Segment Routing, including hands-on work with routers running the Junos OS.

This book has two parts:

- Chapters 1-4 describe the fundamental building blocks of Segment Routing, including a discussion of the different types of segments that exist. In your lab you will build up a network containing various segment types, using the book's examples. The author of these chapters is Julian Lucek.
- Chapter 5 shows you how the fast re-route mechanisms work in SPRING networks, as well as how to configure these mechanisms. The author of this chapter is Krzysztof Szarkowicz.

Preface: A Short History of Segment Routing and Why It Is Popular

Let's begin with a short history of Segment Routing (SR). One of the key properties of SR is that MPLS labels are distributed via the Interior Gateway Protocol (IGP), ISIS or OSPF, rather than a dedicated label distribution protocol. Another is the concept of Source Routing - encoding the path to be taken by a packet by means of a stack of headers applied to the packet when launching it into the network. A draft from Juniper containing these elements got the ball rolling in the IETF in February 2013, and drafts from other vendors soon followed. Later in 2013, an IETF working group dedicated to SR was formed, called "SPRING" (Source Packet Routing in Networking). The terms "Segment Routing" and "SPRING" are synonymous, and are used interchangeably throughout this book. Over the last five years, a myriad of drafts have been published in the IETF covering architecture, use cases, and extensions to protocols, including ISIS, OSPF, BGP, and PCEP.

Some of the key use cases for SR include:

- Traffic Engineering
- Shortest Path Routing
- Local Protection

Traffic Engineering

The path of traffic through a network is controlled by virtue of the stack of labels applied to the packet. The concept of using a stack of packet headers to steer a packet through a network is not entirely new – it was employed in the past in Token Ring and in Automated Network Routing (ANR). This shows that network technology often progresses by building on ideas conceived many years ago.

Traffic engineering (TE) is used to steer traffic around potential traffic hotspots in the network, or in order to create paths that meet particular criteria, such as low latency. Another application of traffic engineering is creating pairs of diverse paths as an ingredient of a path-diverse point-to-point virtual circuit service.

With SR, the transit routers do not hold any state related to a traffic-engineered path. In contrast, with RSVP traffic engineering, a transit router takes part in an RSVP session for each LSP passing through it, so there is some control plane overhead associated with each. The control plane has advantages and disadvantages – the advantage is that the control plane gives more visibility of the status of the LSP. The disadvantage is that there is an upper limit in the number of LSPs that can be sustained by a given transit router. In practice, on modern routers the RSVP scaling limit is very high (on the order in hundreds of thousands), so in some networks it is not an issue, but nevertheless not having an upper limit to be concerned about is an advantage of SR.

In cases where a bandwidth assignment is required for an SR LSP, a controller is needed to perform the path computation and keep track of the bandwidth assignment, as SR has no mechanism to reserve bandwidth at the network layer. In contrast, RSVP can reserve bandwidth at the network layer so a controller is not necessarily needed. On the other hand, some use cases, such as path diversity, require a controller regardless of whether SR or RSVP is used. In any case, the use of controllers in conjunction with TE, whether SR or RSVP, is becoming increasingly popular.

Using SR for TE means that RSVP is not needed in the network. This means there is one less protocol to configure and troubleshoot because the IGP is used instead to carry the necessary MPLS label information.

Given the above pros and cons, some operators prefer SR and others prefer RSVP for traffic engineering, so of course the Junos OS and the NorthStar Controller support both.

Shortest Path Routing

Operators employ shortest path routing in situations where traffic engineering is not required. Shortest path routing using MPLS encapsulation is often used so that core routers do not need to hold a full Internet routing table, or in order to provide tunnels across the core for VPN traffic. Before the advent of SR, LDP was used as the label distribution protocol for shortest path MPLS routing. However, SR can be used instead of LDP to achieve the same goals, which has the following advantages:

One less protocol to configure and troubleshoot, because IGP is used instead of LDP to carry the necessary MPLS label information.

As you will see later, you can arrange for the same label value to be used on each hop of a packet's journey to a given destination node, rather than changing hop-by-hop as with LDP. This makes troubleshooting and understanding forwarding table entries much easier.

In situations where LDP speakers are not directly connected but need to exchange labels, a targeted LDP session is needed, which results in additional control plane overhead. An example is using Remote Loop Free Alternates (RLFA) for local protection. In the LDP case, a targeted LDP session is needed between the point of local repair (PLR) and the PQ-node. When using SR, such a session is not needed as the information required to compute the required labels is already known to the PLR via the IGP.

Local Protection

Over the years, various local protection schemes have been devised under the general banner of IP Fast Reroute. These schemes aim to provide fast protection for IP or LDP traffic, and include Loop Free Alternates (LFA) and RLFA. Depending on the network topology, however, full protection coverage is not always achieved. Sometimes it is not possible for a PLR to find a loop-free alternate path. Segment Routing improves on these with the TI-LFA scheme. This improves the coverage by the use of strict forwarding labels to force traffic along a protection path that otherwise would not be loop-free.

Chapter 1

Principles of Segment Routing

Segment Routing (SR) allows you to create end-to-end paths across a network. A path can consist of multiple building blocks called *segments*, so the end-to-path consists of a chain of such segments. Packets are source-routed – the ingress router applies a stack of packet headers to the packet, each header denoting a successive segment in the packet's journey.

NOTE In this book an MPLS label is used to denote each segment. An alternative scheme would be to use IPv6 headers instead of MPLS headers, but this is beyond the scope of this *Day One* book.

Key Concepts

If you examine the network in Figure 1.1, there are several possible paths that traffic from R1 to R14 could take.

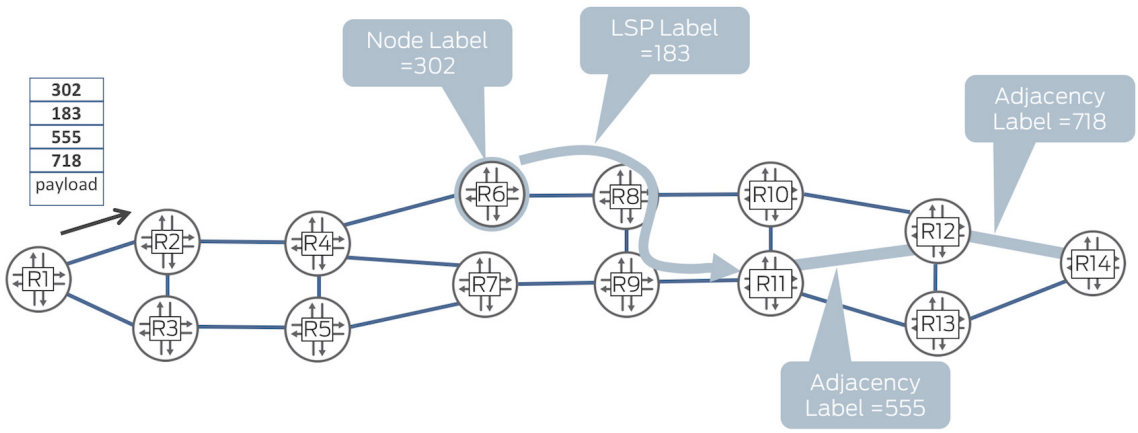

Figure 1.1 *Example Network Showing Different Types of Segments*

Suppose you wish the traffic to be routed as follows:

1. From R1, follow the shortest IGP path to R6.
2. At R6, launch the traffic into the traffic-engineered LSP X. As you can see in Figure 1.1, this LSP follows the path R6, R8, R9, R11.
3. From R11, you want the traffic to use the link to R12.
4. From R12, you want the traffic to use the link to R14.

Each of the four stages in this path has a corresponding segment. An MPLS label is associated with each segment, namely:

- For stage one, the Node Segment is used associated with R6 – in the example the label is 302. This label tells R2 and R4 to forward the packet towards R6 using the shortest IGP path.
- For stage two, the Binding Segment is used associated with TE LSP X. In the example, the label is 183. This label tells R6 to launch the packet onto the LSP

- For stage three, the Adjacency Segment is used associated with the link from R11 to R12. In the example, the label is 555.
- For stage four, the Adjacency Segment is used associated with the link from R12 to R14. In the example, the label is 718.

These illustrate the source-routed nature of SR – R1 pushes the following label stack onto the packet in order to route it through the required sequence of segments: 302 (topmost), 183, 555, and 718. During the course of a packet's journey, the top label is popped when no longer needed. Typically, *penultimate hop popping* (PHP) is used, so, for example, R11 pops label 555 from the packet before forwarding it onto the link to R12, as instructed by that label. In this way, the packet arrives at the destination with just the underlying payload (which may include MPLS VPN labels).

In general, there is a spectrum of behavior that segment routing supports in terms of how much control is exerted on the path that a packet follows through the network. At one end of the spectrum, R1 could send a packet to R14 by simply pushing on the label associated with the node segment of R14. As a result, the packet follows the IGP shortest path to R14. At the other end of the spectrum, the precise hop-by-hop path to be followed by the packet can be engineered by having R1 push onto the packet an adjacency label for each link that the packet must follow in order to reach R14. The example in Figure 1.1 is somewhere in the middle of this spectrum, in that it is specific about some sections of the packet's journey but less specific about other sections.

SR has some neat recursive properties – the traffic-engineered LSP X in Figure 1.1 could be defined in terms of a sequence of adjacency labels. As label 183 is the binding label for that LSP, it is replaced by that sequence of adjacency labels by R6, so in effect a single label is expanded into the multiple labels required to steer the packet hop-by-hop along LSP X.

Later in this book, you will build a network in order to understand how *Adjacency Segments* and *Node Segments* work, including some variations on these themes, such as *Adjacency Sets* and *Anycast Segments*.

Preparing Your Lab Network

Figure 1.2 shows a diagram of the network used in this book. Routers R1 to R9 are MPLS nodes that you will configure to run SR during the course of this book.

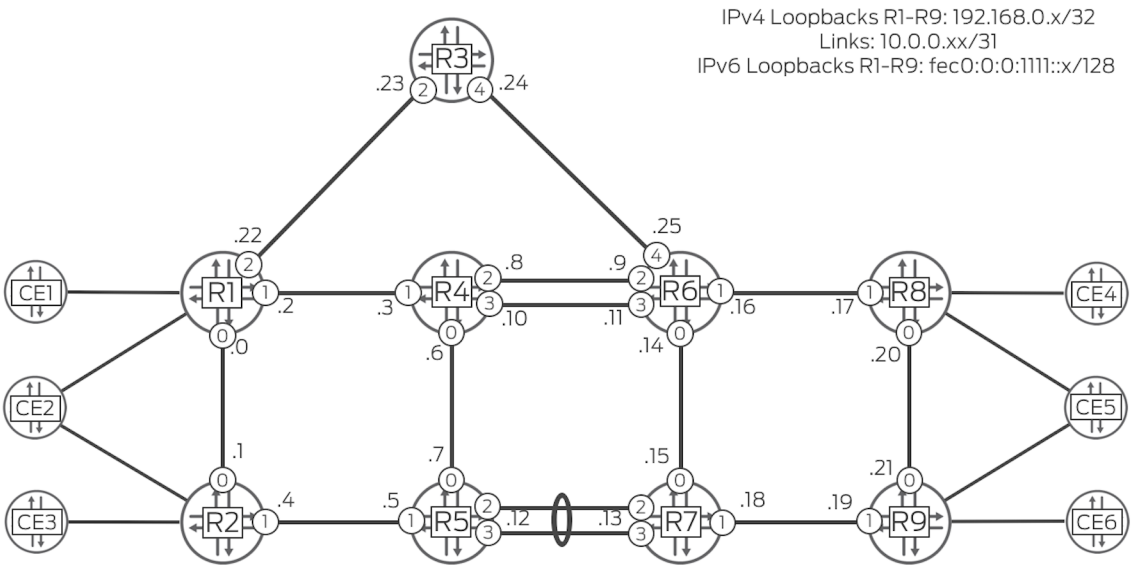

Figure 1.2 *This Book's Lab Diagram*

You can see that CE1 to CE6 are customer edge routers that make use of the services provided by the MPLS network. If you do not have enough routers, you can skip some of the CEs. The examples in this book view the traceroutes between CE3 and CE6, and between CE3 and CE4, so the other CEs are not essential.

The IPv4 loopback addresses of nodes R1 to R9 are of the following form:

- 192.168.0.x/32 for router Rx–for example, 192.168.0.1 for R1, and so on.

The interface addresses are of the following form:

- 10.0.0.y/31 –the value of *y* is shown next to each link on each router, for example R1's address on its interface to R4 is 10.0.0.2.

The IPv6 loopback addresses of nodes R1 to R9 are of the following form:

- fec0:0:0:1111::x/128 for Rx. For example, the loopback address of R2 is fec0:0:0:1111::2/128.

The interface identifiers are of the following form:

- ge-0/0/z –the value of z is shown in a circle on each interface, for example R1's interface to R4 is ge-0/0/1.

Before embarking on the segment routing setup in subsequent chapters, you need to configure a baseline configuration on the network as follows:

- Nodes R1 to R9 are all in ISIS Level 2. Within the ISIS topology, the metric of each link is 1000, except the link between R5 and R7, which is 500 in each direction.
- Family MPLS is configured on all core interfaces.
- One or more MPLS services – you can configure the type of service that interests you most on the routers facing the CE devices (R1, R2, R8, and R9), for example, EVPN or Layer 3 VPN, and so on. For the examples in this book, Layer 3 VPN was configured.
- BGP configuration with relevant address families, given the services that you chose to configure in the previous step. R4 and R7 are router reflectors and R1, R2, R8, and R9 are route reflector clients.

One last thing to note is that the two links between R5 and R7 are in an aggregated Ethernet bundle. In contrast, the two links between R4 and R6 are not.

Chapter 2

Adjacency Segments

This chapter shows you how Adjacency Segments work, using the topology built in the previous chapter. It covers the basic adjacency segment configuration, adjacency sets, and also looks at options for aggregated Ethernet.

Adjacency Segment Configuration

Each router in the topology advertises a label associated with each of its IGP adjacencies with neighboring routers. Such Adjacency Segments have strict forwarding semantics. This means that in the data plane, when a packet arrives with that adjacency label, the router pops it and sends it onto the associated link.

The minimum configuration to invoke Adjacency Segments is to configure the following on all routers from R1 to R9, inclusively:

```
lab@R2# set protocols isis source-packet-routing
```

This triggers each router to dynamically allocate two labels for each adjacency, one for IPv4 and one for IPv6. In order to check this, use the following command on R2:

```
lab@R2> show isis adjacency extensive
R1
  Interface: ge-0/0/0.0, Level: 2, State: Up, Expires in 26 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 15:13:56 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.0
  IPv6 addresses: fe80::5668:a3ff:fe17:3281
  Level 2 IPv4 Adj-SID: 30
  Level 2 IPv6 Adj-SID: 47
  State: Up

R5
  Interface: ge-0/0/1.0, Level: 2, State: Up, Expires in 21 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 15:13:43 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.5
  IPv6 addresses: fe80::5668:a3ff:fe17:312a
  Level 2 IPv4 Adj-SID: 29
  Level 2 IPv6 Adj-SID: 48
  State: Up
```

By way of example, you can see that for its adjacency with R5, router R2 has allocated a label value of 29 for IPv4 and a label value of 48 for IPv6. When you try it yourself, you will probably see different label values, as they are dynamically allocated (`Adj-SID` stands for Adjacency Segment Identifier, meaning the actual label value associated with the Adjacency Segment). Inspection of the mpls.0 table shows the corresponding label operations. For example, when a data-plane packet arrives at R2 with top label 29, the label is popped and the packet is forwarded on ge-0/0/1 (R2's interface facing R5):

```
lab@R2> show route table mpls.0

<snip>

29                 *[L-ISIS/14] 15:13:44, metric 0
                    > to 10.0.0.5 via ge-0/0/1.0, Pop
29(S=0)            *[L-ISIS/14] 00:05:01, metric 0
                    > to 10.0.0.5 via ge-0/0/1.0, Pop
30                 *[L-ISIS/14] 15:13:57, metric 0
                    > to 10.0.0.0 via ge-0/0/0.0, Pop
30(S=0)            *[L-ISIS/14] 00:05:01, metric 0
                    > to 10.0.0.0 via ge-0/0/0.0, Pop
```

```
47                 *[L-ISIS/14] 15:13:57, metric 0
                    > to fe80::5668:a3ff:fe17:3281 via ge-0/0/0.0, Pop
47(S=0)            *[L-ISIS/14] 00:05:01, metric 0
                    > to fe80::5668:a3ff:fe17:3281 via ge-0/0/0.0, Pop
48                 *[L-ISIS/14] 15:13:44, metric 0
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Pop
48(S=0)            *[L-ISIS/14] 00:05:01, metric 0
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Pop
```

Unprotected and Protected Adjacency Segments

By default, an adjacency segment can benefit from fast protection mechanisms, if such mechanisms are configured. These mechanisms will be covered in Chapter 5 of this book. In some cases, you may have some traffic that you want to benefit from fast protection, but other traffic does not need such protection. In order to distinguish between these traffic types, you can arrange for separate adjacency labels to be allocated. Try this on R1:

```
lab@R1# set protocols isis interface ge-0/0/1.0 level 2 ipv4-adjacency-segment protected dynamic

lab@R1# set protocols isis interface ge-0/0/1.0 level 2 ipv4-adjacency-segment unprotected dynamic

lab@R1# set protocols isis interface ge-0/0/1.0 level 2 ipv6-adjacency-segment protected dynamic

lab@R1# set protocols isis interface ge-0/0/1.0 level 2 ipv6-adjacency-segment unprotected dynamic
```

The `dynamic` keyword triggers the router to automatically allocate a label value. Later in this chapter, you will see an example of a statically configured label value. The preceding four commands result in four adjacency labels to be allocated to ge-0/0/1 as follows:

- IPv4 traffic, protection required
- IPv4 traffic, no protection required
- IPv6 traffic, protection required
- IPv6 traffic, no protection required

You can verify this by issuing the following command on R1:

```
lab@R1> show isis adjacency extensive | no-more
R2
  Interface: ge-0/0/0.0, Level: 2, State: Up, Expires in 19 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 15:11:18 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.1
  IPv6 addresses: fe80::5668:a3ff:fe17:31f6
  Level 2 IPv4 Adj-SID: 70
  Level 2 IPv6 Adj-SID: 71
```

```
  State: Up

R4
  Interface: ge-0/0/1.0, Level: 2, State: Up, Expires in 22 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 15:11:11 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.3
  IPv6 addresses: fe80::5668:a3ff:fe17:2509
  Level 2 IPv4   protected Adj-SID:      72, Flags: -BVL--
  Level 2 IPv4 unprotected Adj-SID:      73, Flags: --VL--
  Level 2 IPv6   protected Adj-SID:     102, Flags: FBVL--
  Level 2 IPv6 unprotected Adj-SID:     103, Flags: F-VL--
  State: Up

R3
  Interface: ge-0/0/2.0, Level: 2, State: Up, Expires in 20 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 15:11:33 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.23
  IPv6 addresses: fe80::5668:a3ff:fe17:31a7
  Level 2 IPv4 Adj-SID: 68
  Level 2 IPv6 Adj-SID: 69
  State: Up
```

As you can see, highlighted in bold, separate labels are allocated for protected versus unprotected traffic on the adjacency with R4 (via ge-0/0/1).

The ISIS extensions for segment routing include an Adj-SID sub-TLV to enable a router to advertise its Adj-SIDs throughout the IGP area. In this way all routers from R1 to R9 in the topology are aware of all the Adj-SIDs on all of the links in the topology. To illustrate this, look at the ISIS database. You can do this on any of the routers, as they all have identical ISIS databases. Here is a snippet, showing the Adj-SIDs generated and advertised by R1. As you can see, the information is consistent with what you saw previously:

```
lab@R2> show isis database extensive | no-more
IS-IS level 1 link-state database:

IS-IS level 2 link-state database:

R1.00-00 Sequence: 0x45, Checksum: 0xa038, Lifetime: 63601 secs

<snip>

   IS neighbor: R2.00                          Metric:     1000
     Two-way fragment: R2.00-00, Two-way first fragment: R2.00-00
     P2P IPv4 Adj-SID:      70, Weight:   0, Flags: -BVL--
     P2P IPv6 Adj-SID:      71, Weight:   0, Flags: FBVL--
   IS neighbor: R3.00                          Metric:     1000
     Two-way fragment: R3.00-00, Two-way first fragment: R3.00-00
     P2P IPv4 Adj-SID:      68, Weight:   0, Flags: -BVL--
     P2P IPv6 Adj-SID:      69, Weight:   0, Flags: FBVL--
```

```
   IS neighbor: R4.00                          Metric:     1000
     Two-way fragment: R4.00-00, Two-way first fragment: R4.00-00
     P2P IPv4 Adj-SID:      72, Weight:   0, Flags: -BVL--
     P2P IPv4 Adj-SID:      73, Weight:   0, Flags: --VL--
     P2P IPv6 Adj-SID:     102, Weight:   0, Flags: FBVL--
     P2P IPv6 Adj-SID:     103, Weight:   0, Flags: F-VL--

<snip>
```

The F-flag is the Address Family flag. As you can see, it is set for IPv6 Adj-SIDs and clear for IPv4 Adj-SIDs.

The B-flag is the Backup flag. It is set if the SID is eligible for protection, and clear if it is not. You can see that it is not set for the unprotected label values (73 and 103).

The V-flag is set if the Adj-SID carries a Value, which is the case in all the examples above.

The L-flag is set if the label has Local significance, which is also the case in all the examples above.

The final two flags are not set in any of the examples above - later you will see an example in which they are set.

Mapping the Topology

At this point, as an exercise, let's work out what adjacency label has been allocated on each adjacency in each direction within the ISIS topology. As seen in the previous section, can do this by looking at `show ISIS adjacency extensive` on every router, or looking at `show ISIS database extensive` on any one router. By doing it the latter way, you can satisfy yourself that any randomly chosen router in the ISIS topology has visibility to all of the adjacency labels throughout the network, as these are communicated via ISIS.

Figure 2.1 illustrates the outcome of doing this. Note that to avoid clutter, only the IPv4 adjacency labels are shown. As the labels are dynamically allocated, the label values that you will see in your setup will typically be different. The key point here is that a given node allocates a different label for each of its adjacencies but the same value may get allocated by different nodes. This is perfectly fine, as the labels only have local significance. For example, in Figure 2.1, label 16 happened to get allocated by both R3 and R5.

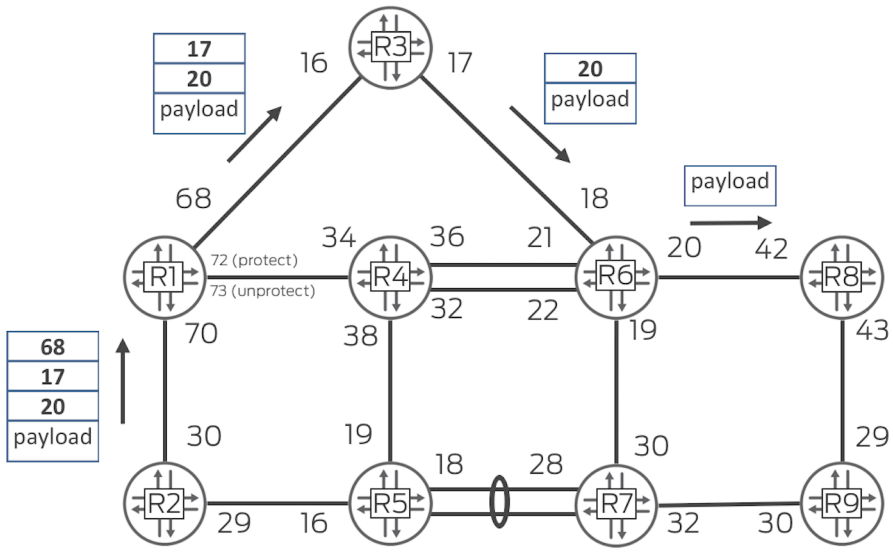

Figure 2.1 *The Network Topology Showing Allocated Adjacency Labels*

NOTE If you're thinking that mapping out the labels is laborious, Juniper's NorthStar Controller does it automatically, by virtue of being part of the control plane of the network.

Suppose R2 wants to send traffic to R8, not via the shortest IGP path (via R5 and R7) but along the path R2-R1-R3-R6-R8. It achieves this by pushing the corresponding stack of adjacency labels onto the packet, as shown in Figure 2.1. Note that in the label stack, there is no label corresponding to the R2-R1 hop, as R2 simply pushes the packet towards R1. R1 sees that the top label is 68, which triggers it to pop the label and forward the packet to R3. In turn, R3 sees that the top label is 17, which triggers it to pop the label and forward the packet to R6. R6 sees label 20, which triggers it to pop the label and forward the packet to R8.

In this book you will see methods to program such label stacks on an ingress node, including static CLI configuration.

Adjacency Set

It's possible to assign a common label to multiple adjacencies on a node. This is known as an *Adjacency Set* (Adj-Set) for short. Let's configure one on R6.

In the previous examples, dynamically-assigned labels were used. In this example, a statically-assigned label is used in order to show the difference in configuration.

First, configure a static label-range like this:

```
lab@R6# set protocols mpls label-range static-label-range 600000 699999
```

Then put two of the interfaces into a set, and assign a corresponding label:

```
set protocols isis interface-group group1 interface ge-0/0/2.0 weight 1
set protocols isis interface-group group1 interface ge-0/0/4.0 weight 3
set protocols isis interface-group group1 level 2 ipv4-adjacency-segment protected label 600001
```

This means that when data-plane packets arrive at R6 with a top label value of 600001, the label is popped and the packets are load-balanced on links ge-0/0/2 and ge-0/0/4 in a 1:3 ratio. You can confirm this like so:

```
lab@R6> show route table mpls.0 detail label 600001

mpls.0: 54 destinations, 54 routes (54 active, 0 holddown, 0 hidden)
600001 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Level: 2
                Next hop type: Router, Next hop index: 0
                Address: 0xb21f310
                Next-hop reference count: 1
                Next hop: 10.0.0.8 via ge-0/0/2.0 weight 0x1 balance 25%
                Label operation: Pop

<snip>
                Next hop: 10.0.0.24 via ge-0/0/4.0 weight 0x1 balance 75%, selected
                Label operation: Pop

<snip>
```

As you can see, the Adj-Set is a convenient way of engineering the way link resources on a given node are utilized. Now look at the ISIS database on any of the routers:

```
lab@R2> show isis database extensive | no-more

<snip>

R6.00-00 Sequence: 0x4d, Checksum: 0xb376, Lifetime: 10703 secs
  IPV4 Index: 406, IPV6 Index: 606
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 9000, Label-Range: [ 1000, 9999 ]
   IS neighbor: R3.00                          Metric:     1000
     Two-way fragment: R3.00-00, Two-way first fragment: R3.00-00
     P2P IPv4 Adj-SID:      18, Weight:   0, Flags: -BVL--
     P2P IPv6 Adj-SID:      24, Weight:   0, Flags: FBVL--
     P2P IPv4 Adj-SID:  600001, Weight:   3, Flags: -BVLSP
   IS neighbor: R4.00                          Metric:     1000
     Two-way fragment: R4.00-00, Two-way first fragment: R4.00-00
     P2P IPv4 Adj-SID:      22, Weight:   0, Flags: -BVL--
     P2P IPv6 Adj-SID:      25, Weight:   0, Flags: FBVL--
     P2P IPv4 Adj-SID:      21, Weight:   0, Flags: -BVL--
```

```
    P2P IPv6 Adj-SID:      23, Weight:   0, Flags: FBVL--
    P2P IPv4 Adj-SID:  600001, Weight:   1, Flags: -BVLSP
  IS neighbor: R7.00                        Metric:     1000
    Two-way fragment: R7.00-00, Two-way first fragment: R7.00-00
    P2P IPv4 Adj-SID:      19, Weight:   0, Flags: -BVL--
    P2P IPv6 Adj-SID:      26, Weight:   0, Flags: FBVL--
  IS neighbor: R8.00                        Metric:     1000
    Two-way fragment: R8.00-00, Two-way first fragment: R8.00-00
    P2P IPv4 Adj-SID:      20, Weight:   0, Flags: -BVL--
    P2P IPv6 Adj-SID:      27, Weight:   0, Flags: FBVL--

<snip>
```

As you can see, the Adj-Set label value of 600001, and associated weight, is advertised in conjunction with the associated adjacencies (in addition to the regular Adj-SIDs). The S-Flag means that the SID refers to a set (*S*-) of adjacencies. The P-Flag means that the Adj-SID is persistently (*P*-) allocated (because it was statically configured) and therefore remains consistent across router restarts and/or interface flaps.

It's worth pausing here to discuss *statically allocated* versus *dynamically allocated* labels. Statically allocated labels are useful in conjunction with statically configured traffic-engineered SR LSPs – configuration of these involves explicitly listing on the ingress node the label values to be used in the stack. If dynamic labels were used, the configuration on the ingress node would need to be changed each time a label changes. On the other hand, if the label stack is installed by a controller that has visibility of the label values used in the network, then dynamically allocated labels are more convenient as they involve less configuration.

Label Per Member Link on Aggregated Ethernet

Let's have a look at Adjacency Labels in the context of aggregated Ethernet, for example on R5:

```
lab@R5> show isis adjacency detail
R7
  Interface: ae0.0, Level: 2, State: Up, Expires in 24 secs
  Priority: 0, Up/Down transitions: 11, Last transition: 2d 01:08:27 ago
  Circuit type: 2, Speaks: IP, IPv6
  Topologies: Unicast
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 10.0.0.13
  IPv6 addresses: fe80::2e6b:f5ff:fe2e:70c0
  Level 2 IPv4 Adj-SID: 18, IPv6 Adj-SID: 22

<snip>
```

Given that the ISIS adjacency is at the ae0.0 level, the Adj-SID is at that level, too. This means that if packets arrive at R5 with label 18 (the IPv4 Adj-SID for the ae0.0 link), they are load-balanced across the child links of ae0.0 on a per-flow basis.

In some cases, you might want to nail a flow of packets to a particular child link. You can achieve this by allocating a separate label per member link of the aggregated interface (while still having an Adj-SID for the aggregated interface as a whole). These labels per member link are statically configured, and are not advertised in the IGP. A use case for this is when a Network Management System (NMS) needs to send probe packets through the network to verify that a link is truly up from the forwarding plane point of view.

Try it yourself. First, allocate a static label range on R7:

```
set protocols mpls label-range static-label-range 700000 799999
```

Then, allocate static labels to each of the member links of the aggregated link on R7 as follows:

```
set protocols mpls static-label-switched-path ae0-ge-0-0-2 transit 700002 next-hop 10.0.0.12

set protocols mpls static-label-switched-path ae0-ge-0-0-2 transit 700002 member-interface ge-0/0/2

set protocols mpls static-label-switched-path ae0-ge-0-0-2 transit 700002 pop

set protocols mpls static-label-switched-path ae0-ge-0-0-3 transit 700003 next-hop 10.0.0.12

set protocols mpls static-label-switched-path ae0-ge-0-0-3 transit 700003 member-interface ge-0/0/3

set protocols mpls static-label-switched-path ae0-ge-0-0-3 transit 700003 pop
```

This results in static label-switched path stanzas like this:

```
user@R7> show configuration protocols mpls
<snip>
static-label-switched-path ae0-ge-0-0-2 {
    transit 700002 {
        next-hop 10.0.0.12;
        member-interface ge-0/0/2;
        pop;
    }
}
static-label-switched-path ae0-ge-0-0-3 {
    transit 700003 {
        next-hop 10.0.0.12;
        member-interface ge-0/0/3;
        pop;
    }
}
<snip>
```

The configuration means that when a packet arrives at R7 with top label 700002, the label is popped and the packet is sent out on member link ge-0/0/2 of ae0. On the other hand, when a packet arrives at R7 with top label 700003, the label is popped and the packet is sent out on member link ge-0/0/3 of ae0.

You can verify this by looking at the mpls.0 table on R7:

```
user@R7> show route table mpls.0 label 700002 detail | no-more

mpls.0: 45 destinations, 45 routes (45 active, 0 holddown, 0 hidden)
700002 (1 entry, 1 announced)
        *MPLS   Preference: 6
                Next hop type: Router, Next hop index: 599
                Address: 0xb3a28f0
                Next-hop reference count: 2
                Next hop: 10.0.0.12 via ae0.0 -> ge-0/0/2 weight 0x1, selected
                Label operation: Pop
<snip>


user@R7> show route table mpls.0 label 700003 detail | no-more

mpls.0: 45 destinations, 45 routes (45 active, 0 holddown, 0 hidden)
700003 (1 entry, 1 announced)
        *MPLS   Preference: 6
                Next hop type: Router, Next hop index: 602
                Address: 0xb3a2a70
                Next-hop reference count: 2
                Next hop: 10.0.0.12 via ae0.0 -> ge-0/0/3 weight 0x1, selected
                Label operation: Pop
<snip>
```

Summary

This chapter has shown you how to configure and examine Adjacency Seg-ments, including Adjacency Sets, which are a key ingredient of traffic engineering using Segment Routing.

Chapter 3

Node Segments and Anycast Segments

In this chapter, you will configure and use Node Segments. These are used for shortest-path forwarding using MPLS. You might ask, *If LDP already does this job, why use Node Segments instead?* Well, Node Segments have some key advantages over LDP that are discussed within the chapter. You will also configure Anycast Segments, which are a useful way of steering traffic through a particular region of the network.

Node Segment Principles

Each node in the network has a node segment associated with its loopback address. Any other node in the network can send packets to it along the shortest IGP path by using that node segment. By way of example, let's have a look at Figure 3.1.

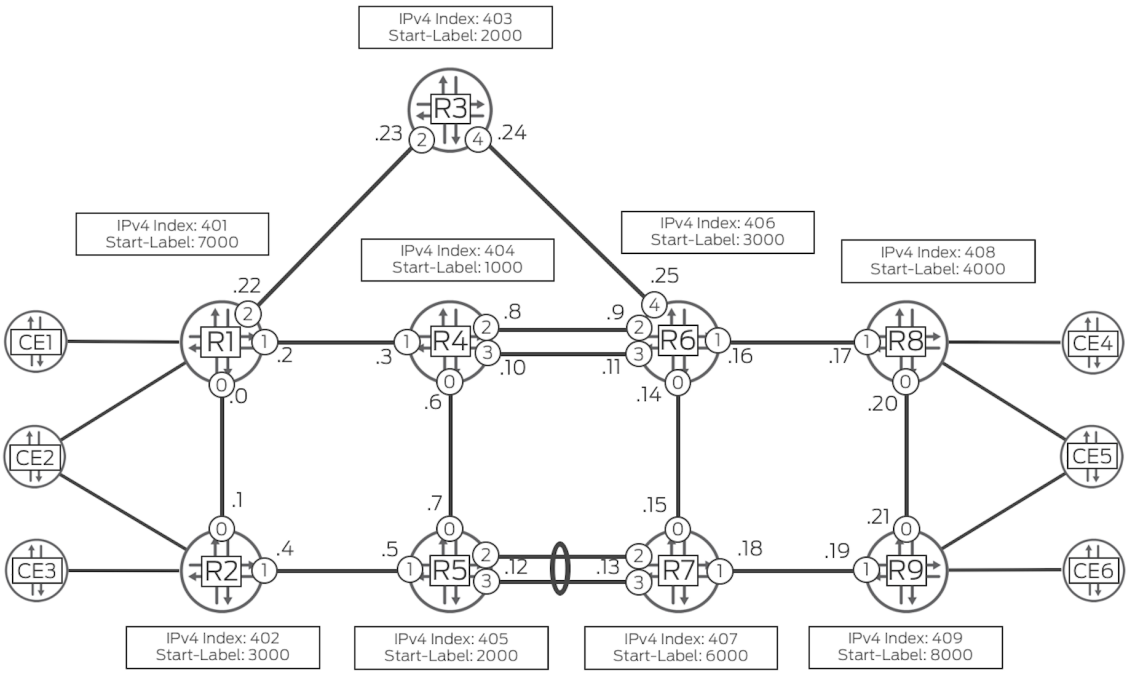

Figure 3.1 *Network Diagram Illustrating Node Segment Operation*

R2 wants to send packets to R6 using the shortest IGP path. It identifies its shortest path neighbor towards R6, namely R5. It pushes the MPLS label onto the packet that R5 associates with R6's node segment and sends it on the link to R5. R5 in turn swaps that label for the label that its shortest path neighbor, R7, associates with R6, and sends the packet to R7. R7 pops the label and sends the underlying packet to R6.

But how does each router involved know what label its shortest path neighbor is expecting to see in order to forward to R6? In order to achieve this, each router is configured with two key pieces of information:

- *A node index*: Each router must have a unique node index. This is also known as a Node-SID (Node Segment Identifier).
- *A label block*: This is defined in terms of a start-label and a label-range. The label range must be wide enough to accommodate all of the routers in the domain (including anticipated future growth). This label range is known as the segment routing global block (SRGB).

The boxes in Figure 3.1 show the node-index and start-label for each of the routers in the network (the end of the label-range is not shown in the diagram to avoid clutter). Notice that each node label is unique. However, the label-ranges can overlap – for example, R3 happens to use the same start-label as R5. The node-index and label-range information is distributed in the IGP, so all of the routers in the network are aware of it. Let's now return to our example of R2 sending packets to destination R6 along the shortest path. R2 calculates what label its shortest path neighbor R5 associates with R6's node segment. This is calculated as follows:

```
Start-label of R5's label range + index advertised by R6

2000 + 406 = 2406
```

R2 pushes label 2406 onto the packet and forwards it to R5. R5 swaps label 2406 for the label that R7 associates with R9's node segment (6000 + 406 = 6406) and sends it to R7. R7 pops the label and sends the rest of the packet to R6.

This is a generic example in which the routers involved have different label ranges allocated for node segments. Now look at Figure 3.2. In this example, each router is configured with the *same* start-label, 1000, and it has a very useful effect: the label required to reach R6 via the shortest path is the same at each hop in the network (1000 + 406 = 1406). This makes troubleshooting much easier– when looking at routing or forwarding tables on any router in the network, label 1406 means shortest path forwarding to R6.

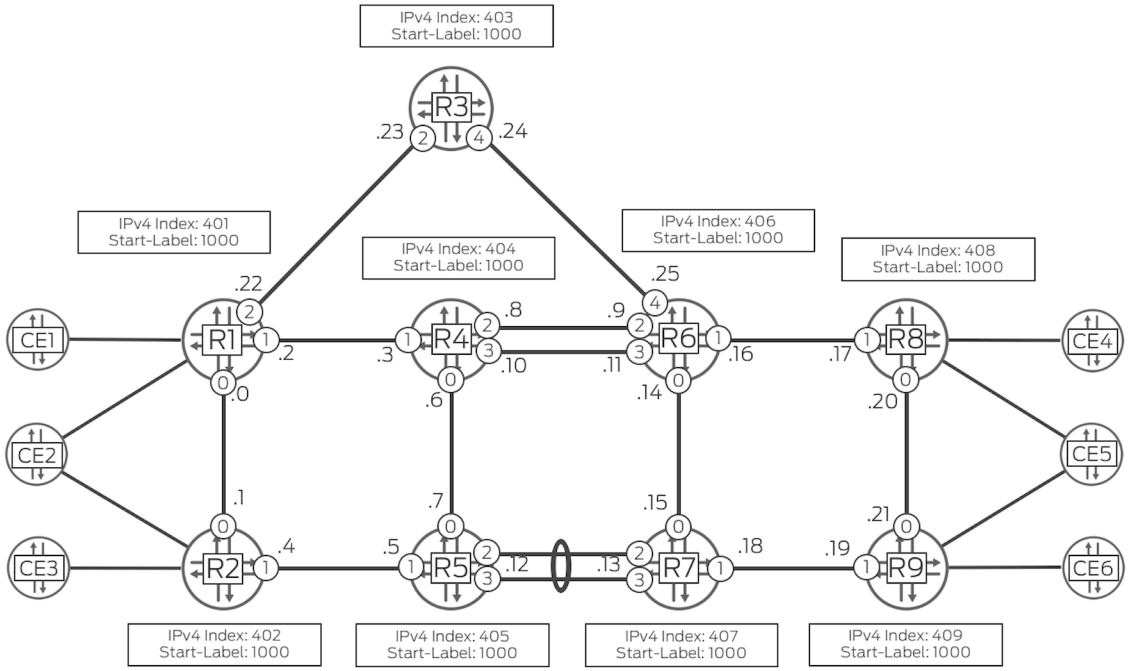

Figure 3.2 *Node Segment Operation with Same Label-range on Each Node*

Okay, now let's configure the network using the same label-range on each router, a start-label of 1000 and a range of 9000. An additional property of node segments not mentioned in Figure 3.2's example is that node segments can be used for IPv6 traffic as well as IPv4 traffic. In that case, separate indices are configured on each router for IPv6 versus IPv4. Let's use the following scheme. The index for router Rx is 40x for IPv4 and 60x for IPv6. Configure this on each of the routers in the topology. For example, on R1, you configure the following:

```
set protocols isis source-packet-routing srgb start-label 1000
set protocols isis source-packet-routing srgb index-range 9000
set protocols isis source-packet-routing node-segment ipv4-index 401
set protocols isis source-packet-routing node-segment ipv6-index 601
```

On R2, configure the following:

```
set protocols isis source-packet-routing srgb start-label 1000
set protocols isis source-packet-routing srgb index-range 9000
set protocols isis source-packet-routing node-segment ipv4-index 402
set protocols isis source-packet-routing node-segment ipv6-index 602
```

And so on, for the other routers R3-R9.

You can confirm these settings on each router as follows:

```
lab@R2> show isis overview | no-more
Instance: master
  Router ID: 192.168.0.2
  Hostname: R2
  Sysid: 1921.6800.0002
  Areaid: 49.0000
  Adjacency holddown: enabled
  Maximum Areas: 3
  LSP life time: 65535
  Reference bandwidth: 1000000000000
  Attached bit evaluation: enabled
  SPF delay: 50 msec, SPF holddown: 2000 msec, SPF rapid runs: 5
  Overload bit at startup is set
    Overload high metrics: disabled
    Allow route leaking: disabled
    Overload timeout: 300 sec
  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 : 9000
    SRGB Block Allocation: Success
      SRGB Start Index : 1000, SRGB Size : 9000, Label-Range: [ 1000, 9999 ]
    Node Segments: Enabled
      Ipv4 Index : 402, Ipv6 Index : 602
  Post Convergence Backup: Disabled
  Level 1
    Internal route preference: 15
    External route preference: 160
    Prefix export count: 0
    Wide metrics are enabled, Narrow 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
    Source Packet Routing is enabled
```

The index and label-range information is flooded by ISIS throughput the topology, so all the routers see the information. You can verify this by looking at the ISIS database on any of the routers. For example, by looking at R7's ISIS database, you can see the IPv4 index and IPv6 index advertised by R2, together with its label-block:

```
lab@R7> show isis database R2.00-00 detail
IS-IS level 1 link-state database:

IS-IS level 2 link-state database:

R2.00-00 Sequence: 0x4c, Checksum: 0x369c, Lifetime: 48138 secs
  IPV4 Index: 402, IPV6 Index: 602
```

```
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 9000, Label-Range: [ 1000, 9999 ]

<snip>
```

Let's now look at the inet.3 table on router R2:

```
lab@R2> show route table inet.3

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

192.168.0.1/32     *[L-ISIS/14] 1w0d 19:36:50, metric 1000
                    > to 10.0.0.0 via ge-0/0/0.0
192.168.0.3/32     *[L-ISIS/14] 1w0d 19:36:50, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1403
192.168.0.4/32     *[L-ISIS/14] 1w0d 19:36:50, metric 2000
                       to 10.0.0.0 via ge-0/0/0.0, Push 1404
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1404
192.168.0.5/32     *[L-ISIS/14] 1w0d 19:36:50, metric 1000
                    > to 10.0.0.5 via ge-0/0/1.0
192.168.0.6/32     *[L-ISIS/14] 1d 15:17:22, metric 2500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1406
192.168.0.7/32     *[L-ISIS/14] 1d 15:17:22, metric 1500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1407
192.168.0.8/32     *[L-ISIS/14] 1d 15:17:22, metric 3500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1408
192.168.0.9/32     *[L-ISIS/14] 1d 15:17:22, metric 2500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1409
```

As you can see, the table contains an entry for the loopback address of each of the other routers in the topology. The protocol L-ISIS stands for *labeled* ISIS.

Let's focus first of all on the entry to 192.168.0.6 (the loopback address of R6). R2 has determined that its shortest path neighbor to R6 is R5. It has added the starting value of the label-range advertised by R5 (1000) to the index advertised by R6 (406) to give a label value of 1406. Hence the action is to push label 1406 onto the packet and push it out on the link to R5, ge-0/0/1. If you now look at the corresponding entry in the mpls.0 table on R5, you see that it swaps the incoming label value of 1406 to an outgoing label value of 1406, because R5's shortest path neighbor to R6, namely R7, is also configured with 1000 as the starting value of its label-range:

```
lab@R5> show route table mpls.0 label 1406

mpls.0: 42 destinations, 42 routes (42 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1406               *[L-ISIS/14] 1d 16:11:01, metric 1500
                    > to 10.0.0.13 via ae0.0, Swap 1406
```

This illustrates that if you configure the same SRGB on each node, the label used to reach a given node is the same on each hop. It makes things very convenient for troubleshooting.

Let's now look at an ECMP case. R2 has two ECMP paths to R4, via R1 and R5. Looking again at R2's inet.3 table above, you see two next hops to R4's loopback, 192.168.0.4 the link to R1, and the link to R5. In either case, the label operation is to push label 1404 onto the packet.

Now let's ook at how routes resolve over entries in inet.3. As always, by default, routes whose BGP next hop matches an entry in inet.3, resolve over the inet.3 entry. By way of an example, look at an L3VPN prefix:

```
lab@R2> show route table RI-L3VPN-1.inet.0

<snip>

172.16.3.4/32      *[BGP/170] 00:15:57, localpref 100, from 192.168.0.4
                      AS path: 65301 I, validation-state: unverified
                    > to 10.0.0.5 via ge-0/0/1.0, Push 34, Push 1408(top)
                    [BGP/170] 00:38:33, localpref 100, from 192.168.0.7
                      AS path: 65301 I, validation-state: unverified
                    > to 10.0.0.5 via ge-0/0/1.0, Push 34, Push 1408(top)
<snip>
```

The prefix 172.16.3.4/32 shown here belongs to the RI-L3VPN-1 VRF. It belongs to CE4 and was advertised by R8 (the output does not explicitly show this, but you can verify it using the `detail` version of the command). It is received by R2 via the route reflectors R4 and R7. Packets sent to R8 have a VPN label value 34 pushed onto them, and then the node label, 1408, corresponding to R8 on top. You can verify this by doing a traceroute from CE3 to that address:

```
lab@CE3> traceroute 172.16.3.4
traceroute to 172.16.3.4 (172.16.3.4), 30 hops max, 52 byte packets
 1  10.3.1.2 (10.3.1.2)  42.119 ms  63.853 ms  62.910 ms
 2  10.0.0.5 (10.0.0.5)  117.026 ms  87.713 ms  82.022 ms
     MPLS Label=1408 CoS=0 TTL=1 S=0
     MPLS Label=34 CoS=0 TTL=1 S=1
 3  10.0.0.13 (10.0.0.13)  111.287 ms  6.289 ms  5.795 ms
     MPLS Label=1408 CoS=0 TTL=1 S=0
     MPLS Label=34 CoS=0 TTL=2 S=1
 4  10.0.0.19 (10.0.0.19)  6.078 ms 10.0.0.14 (10.0.0.14)  10.324 ms  5.639 ms
     MPLS Label=1408 CoS=0 TTL=1 S=0
     MPLS Label=34 CoS=0 TTL=3 S=1
 5  10.3.1.250 (10.3.1.250)  5.658 ms  5.811 ms  7.861 ms
 6  172.16.3.4 (172.16.3.4)  6.619 ms  6.544 ms  7.370 ms
```

As expected, the node-label value of 1408 is the same on each hop and so far we have been looking at IPv4 node-labels. Let's now turn our attention to IPv6 node-labels, by looking at the inet6.3 table on R2:

```
lab@R2> show route table inet6.3

inet6.3: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::ffff:192.168.0.1/128
                   *[L-ISIS/14] 15:13:57, metric 1000
```

```
                    > to 10.0.0.0 via ge-0/0/0.0
::ffff:192.168.0.3/128
                   *[L-ISIS/14] 15:13:57, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1403
::ffff:192.168.0.4/128
                   *[L-ISIS/14] 15:13:44, metric 2000
                      to 10.0.0.0 via ge-0/0/0.0, Push 1404
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1404
::ffff:192.168.0.5/128
                   *[L-ISIS/14] 15:13:44, metric 1000
                    > to 10.0.0.5 via ge-0/0/1.0
::ffff:192.168.0.6/128
                   *[L-ISIS/14] 15:10:00, metric 2500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1406
::ffff:192.168.0.7/128
                   *[L-ISIS/14] 15:13:44, metric 1500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1407
::ffff:192.168.0.8/128
                   *[L-ISIS/14] 15:10:00, metric 3500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1408
::ffff:192.168.0.9/128
                   *[L-ISIS/14] 15:10:00, metric 2500
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1409
::ffff:192.168.0.99/128
                   *[L-ISIS/14] 15:13:44, metric 2000
                      to 10.0.0.0 via ge-0/0/0.0, Push 1499
                    > to 10.0.0.5 via ge-0/0/1.0, Push 1499
fec0:0:0:1111::1/128
                   *[L-ISIS/14] 15:13:57, metric 1000
                    > to fe80::5668:a3ff:fe17:3281 via ge-0/0/0.0
fec0:0:0:1111::3/128
                   *[L-ISIS/14] 15:13:57, metric 2000
                    > to fe80::5668:a3ff:fe17:3281 via ge-0/0/0.0, Push 1603
fec0:0:0:1111::4/128
                   *[L-ISIS/14] 15:13:44, metric 2000
                      to fe80::5668:a3ff:fe17:3281 via ge-0/0/0.0, Push 1604
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1604
fec0:0:0:1111::5/128
                   *[L-ISIS/14] 15:13:44, metric 1000
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0
fec0:0:0:1111::6/128
                   *[L-ISIS/14] 15:10:00, metric 2500
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1606
fec0:0:0:1111::7/128
                   *[L-ISIS/14] 15:13:44, metric 1500
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1607
fec0:0:0:1111::8/128
                   *[L-ISIS/14] 15:10:00, metric 3500
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1608
fec0:0:0:1111::9/128
                   *[L-ISIS/14] 15:10:00, metric 2500
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1609
```

As you can see, the table contains an entry for the IPv6 loopback address for each of the other routers in the topology. Let's focus on the entry to fec0:0:0:1111::6/128 (the IPv6 loopback address of R6). R2 has determined that its shortest path neighbor to R6 is R5. It has added the starting value of the

label-range advertised by R5 (1000) to the index advertised by R6 (606) to give a label value of 1606. Hence the action is to push label 1606 onto the packet and push it out on the link to R5, ge-0/0/1.

Now let's look at how routes resolve entries in inet6.3. As always, by default, routes whose BGP next hop matches an entry in inet6.3 resolve over the inet6.3 entry. By way of an example, let's look at an IPv6 prefix. The example shows an IPv6 prefix in the inet6.0 table, originated by R8:

```
lab@R2> show route fec0:0:0:8888::8/128

inet6.0: 15 destinations, 16 routes (15 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

<snip>

fec0:0:0:8888::8/128
                   *[BGP/170] 1d 16:16:50, localpref 100, from fec0:0:0:1111::4
                      AS path: I, validation-state: unverified
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1608
                    [BGP/170] 1d 16:16:50, localpref 100, from fec0:0:0:1111::7
                      AS path: I, validation-state: unverified
                    > to fe80::5668:a3ff:fe17:312a via ge-0/0/1.0, Push 1608
```

The prefix fec0:0:0:8888::8/128 is originated by R8 (the output does not explicitly show this, but you can verify this using the `detail` version of the command). In the lab setup, this can be achieved by having a static route to discard on R8 and redistributing to BGP. It is received by R2 via route reflectors R4 and R7. Packets destined to this IPv6 prefix have the IPv6 node-label, value 1608, corresponding to R8, pushed onto them.

Advantages of Using Node Segments Instead of LDP

Let's now discuss the two key advantages of using Node Segments instead of LDP.

1. You have already seen the first advantage – the fact that when configuring the same SRGB on each router, the label required to reach a given router is the same throughout the network, rather than changing hop-by-hop as in the LDP case.

2. You have fewer protocols to configure and monitor. LDP can be removed from the network.

Anycast Segments

So far in this chapter, we've looked at cases in which each router had a unique SID, a node SID associated with its loopback address. A node SID is actually a special case of a Prefix SID because a Prefix SID is any prefix associated with a node.

One application of Prefix SIDs is to achieve anycast operation. You can ascribe the same IP address, and associated Anycast-SID, to multiple nodes. If in the data plane a packet arrives at a node with the anycast label on top, the node forwards the packet towards the nearest member of the group. In typical deployments, routers in an anycast group would advertise regular node SIDs as well.

You configure an Anycast-SID by applying the following configuration to routers R3 and R4:

```
set interfaces lo0 unit 0 family inet address 192.168.0.99/32
set policy-options policy-statement ISIS-EXPORT term ANYCAST from protocol direct
set policy-options policy-statement ISIS-EXPORT term ANYCAST from interface lo0.0
set policy-options policy-statement ISIS-EXPORT term ANYCAST from route-filter 192.168.0.99/32 exact
set policy-options policy-statement ISIS-EXPORT term ANYCAST then prefix-segment index 499
set policy-options policy-statement ISIS-EXPORT term ANYCAST then accept
set protocols isis export ISIS-EXPORT
```

The effect of this configuration is that both R3 and R4 advertise the prefix 192.168.0.99 and associated index of 499. The label-range (SRGB) used is the same as you previously configured when applying the node segment configuration. Given that you configured each router in the topology with a value of 1000 as the start of its SRGB, the label required to reach the anycast group is 1000 + 499 = 1499, wherever you are in the network.

Have a look at R3 and R4 in the ISIS database of any of the nodes in the topology. Here is the one corresponding to R3, the one corresponding to R4 is similar:

```
lab@R2> show isis database R3.00-00 extensive
IS-IS level 1 link-state database:

IS-IS level 2 link-state database:

R3.00-00 Sequence: 0x48, Checksum: 0x13be, Lifetime: 31955 secs
  IPV4 Index: 403, IPV6 Index: 603
  Node Segment Blocks Advertised:
    Start Index : 0, Size : 9000, Label-Range: [ 1000, 9999 ]


<snip>

  TLVs:
    Area address: 49.0000 (3)
    LSP Buffer Size: 1492
    Speaks: IP
    Speaks: IPV6
    IP router id: 192.168.0.3
    IP address: 192.168.0.3
    Hostname: R3
    IP extended prefix: 192.168.0.99/32 metric 0 up
      8 bytes of subtlvs
      Prefix SID, Flags: 0x00(R:0,N:0,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 499
    IP extended prefix: 192.168.0.3/32 metric 0 up
      8 bytes of subtlvs
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 403
```

```
    IP extended prefix: 10.0.0.22/31 metric 1000 up
    IP extended prefix: 10.0.0.24/31 metric 1000 up
    IP address: 192.168.0.99
    IPv6 prefix: fec0::/32 Metric 0 Up
    IPv6 prefix: fec0:0:0:1111::3/128 Metric 0 Up
      8 bytes of subtlvs
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 603
    Router Capability:  Router ID 192.168.0.3, Flags: 0x00
      SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 9000, SID-Label: 1000
      SPRING Algorithm - Algo: 0
<snip>
```

As you can see, a Prefix SID is advertised associated with the anycast address 192.168.0.99/32, along with the index value of 499.

Now let's look at the mpls.0 table on R2 in the context of this Anycast-SID. As you can see, R2 has two ECMP paths to this anycast region – via R1 (ge-0/0/0) and R5 (ge-1/0/0):

```
root@R2> show route table mpls.0 label 1499

<snip>

1499               *[L-ISIS/14] 00:05:44, metric 2000
                      to 10.0.0.0 via ge-0/0/0.0, Swap 1499
                    > to 10.0.0.5 via ge-0/0/1.0, Swap 1499
```

R5 is directly connected to R4, one of the two members of the anycast group. As you can see, R5 pops the label and sends the traffic to R4 via ge-0/0/0:

```
root@R5> show route table mpls.0 label 1499

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

1499               *[L-ISIS/14] 00:43:42, metric 1000
                    > to 10.0.0.6 via ge-0/0/0.0, Pop
1499(S=0)          *[L-ISIS/14] 00:02:48, metric 1000
                    > to 10.0.0.6 via ge-0/0/0.0, Pop
```

Let's now see what happens on R1. R1 is directly connected to both members of the anycast group, R4 and R3, and the metric to each is the same. As you can see, R1 pops the label and ECMPs the traffic to R4 (via ge-0/0/1) and R3 (via ge-0/0/2):

```
root@R1> show route table mpls.0 label 1499

mpls.0: 61 destinations, 61 routes (61 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1499               *[L-ISIS/14] 1d 06:41:49, metric 1000
                      to 10.0.0.3 via ge-0/0/1.0, Pop
                    > to 10.0.0.23 via ge-0/0/2.0, Pop
1499(S=0)          *[L-ISIS/14] 00:10:42, metric 1000
                      to 10.0.0.3 via ge-0/0/1.0, Pop
                    > to 10.0.0.23 via ge-0/0/2.0, Pop
```

Summary

This chapter has shown how Node Segments are used to achieve shortest-path routing across the network. You also saw how Anycast Segments can be used to steer packets via particular regions of the network. In the next chapter, you will see how interworking between Segment Routing and LDP is achieved.

Chapter 4

Interworking Between Segment Routing and LDP

Chapter 3 looked at how shortest path forwarding can be achieved using Node Segments. In this chapter, you will look at the interworking between that approach and LDP, a useful scenario when the routers in some part(s) of the network do not support SR, but do support LDP.

Note that this scheme is not needed in order to *migrate* from LDP to SR – such migration can be achieved in a way similar to migration from one IGP to another. Rather, it is aimed at scenarios in which some of the routers in the network do not support SR and will continue to use LDP for the foreseeable future.

For this chapter, we will use a new topology shown in Figure 4.1 that illustrates a scenario in which R8 and R9 do not support SR, but do support LDP. Routers R1, R2, R3, and R4 will not run LDP, only SR. R6, at the border between the two regions, will run both LDP and SR. R6 will be responsible for swapping MPLS labels from LDP to SR node labels and vice versa, in order to allow traffic to travel between the two regions.

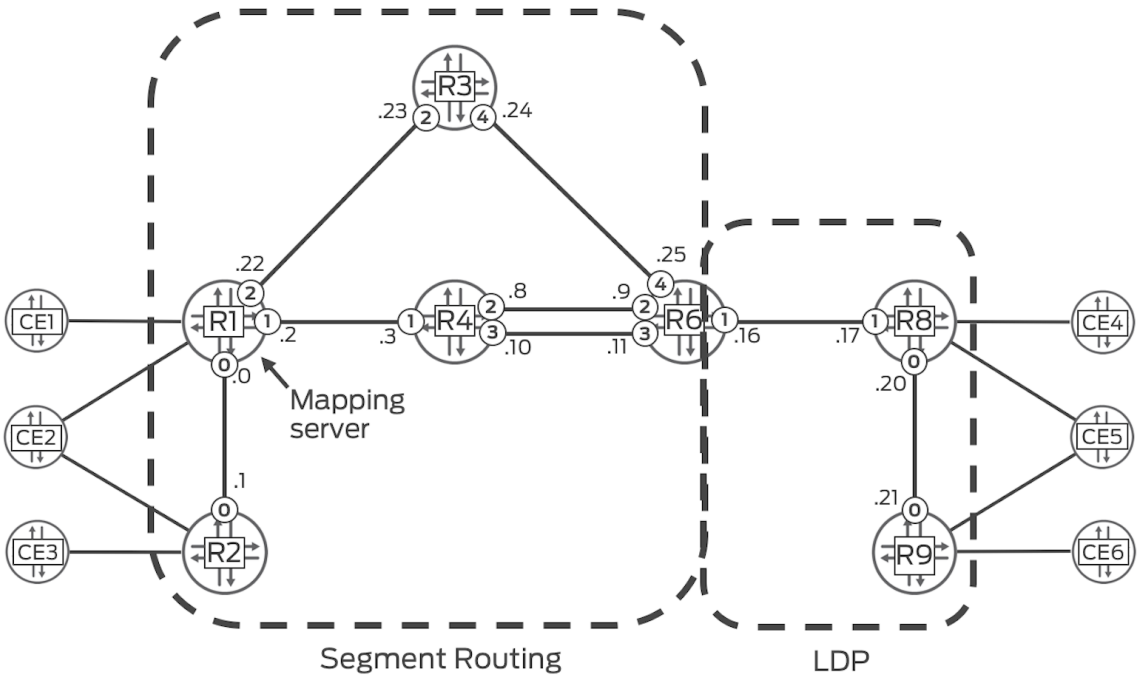

Figure 4.1 *Segment Routing and LDP Interworking Scenario*

Network Setup

First of all, the network should be configured as follows:

1. Use the same configuration on R1, R2, R3, R4 as used in Chapter 3.
2. On R6, use the same configuration as used in Chapter 3, and in addition configure LDP on the loopback interface of R6, and on the interface facing R8.
3. On R8 and R9, use the same configuration as in Chapter 3 except for the following changes:

- Remove all SR configuration from R8 and R9
- Configure LDP on R8 and R9, including the interfaces between R8 and R9 and the interface on R8 facing R6.

Interworking Mechanism

Now let's have a look at the LDP database of R6. As you can see, at this point LDP labels only exist for the LDP speakers, R6, R8, and R9.

```
user@R6> show ldp database
Input label database, 192.168.0.6:0--192.168.0.8:0
Labels received: 3
  Label     Prefix
     55      192.168.0.6/32
      3      192.168.0.8/32
     54      192.168.0.9/32

Output label database, 192.168.0.6:0--192.168.0.8:0
Labels advertised: 3
  Label     Prefix
      3      192.168.0.6/32
     26      192.168.0.8/32
     27      192.168.0.9/32
```

Now, add the following configuration to R6:

```
lab@R6# set protocols ldp sr-mapping-client
```

This triggers R6 to allocate LDP labels corresponding to the node SIDs of R1, R2, R3, and R4. It advertises them to its LDP neighbors (R8 in the example network). For example, R6 allocates a label of value 29 corresponding to the loopback address of R2, 192.168.0.2. (you'll also notice that an LDP label has been allocated for the 192.168.0.99 anycast address that was configured in Chapter 3):

```
user@R6> show ldp database
Input label database, 192.168.0.6:0--192.168.0.8:0
Labels received: 8
  Label     Prefix
     56      192.168.0.1/32
     57      192.168.0.2/32
     58      192.168.0.3/32
     59      192.168.0.4/32
     55      192.168.0.6/32
      3      192.168.0.8/32
     54      192.168.0.9/32
     60      192.168.0.99/32

Output label database, 192.168.0.6:0--192.168.0.8:0
Labels advertised: 8
  Label     Prefix
     28      192.168.0.1/32
     29      192.168.0.2/32
     30      192.168.0.3/32
     31      192.168.0.4/32
      3      192.168.0.6/32
     26      192.168.0.8/32
     27      192.168.0.9/32
     32      192.168.0.99/32
```

Now let's have a look at the mpls.0 table on R6. As you can see below, a label swap operation has been installed, swapping the LDP label value of 29 corresponding to R2 that we saw above for the node-SID value of 1402 corresponding to R2:

```
user@R6> show route table mpls.0

<snip>

29                 *[LDP/9] 00:03:19, metric 1
                    > to 10.0.0.8 via ge-0/0/2.0, Swap 1402
                      to 10.0.0.10 via ge-0/0/3.0, Swap 1402
                      to 10.0.0.24 via ge-0/0/4.0, Swap 1402

<snip>
```

Now look at the inet.3 table on R9. R9 (or indeed, R8) is not explicitly aware of the fact that R6 is performing the LDP to SR translation, as the inet.3 table on R9 looks no different to a situation in which LDP is turned on everywhere and no SR is present in the network – LDP entries are present for each of the other routers in the network:

```
user@R9> show route table inet.3

inet.3: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.1/32     *[LDP/9] 00:07:53, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 56
192.168.0.2/32     *[LDP/9] 00:07:53, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 57
192.168.0.3/32     *[LDP/9] 00:07:53, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 58
192.168.0.4/32     *[LDP/9] 00:07:53, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 59
192.168.0.6/32     *[LDP/9] 00:10:25, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 55
192.168.0.8/32     *[LDP/9] 00:18:56, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0
192.168.0.99/32    *[LDP/9] 00:07:53, metric 1
                    > to 10.0.0.20 via ge-0/0/0.0, Push 60
```

You have now achieved LDP to SR forwarding. Now let's look at forwarding in the opposite direction, from the SR domain to the LDP domain.

Suppose R2 wants to send MPLS traffic to R9, using shortest path forwarding. At this point, R9 does not have the required label information, as you can see by looking at the inet.3 table. In fact, you can only see entries corresponding to the SR speakers in the network:

```
user@R2> show route table inet.3

inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
192.168.0.1/32     *[L-ISIS/14] 00:34:24, metric 1000
                    > to 10.0.0.0 via ge-0/0/0.0
192.168.0.3/32     *[L-ISIS/14] 00:32:26, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1403
192.168.0.4/32     *[L-ISIS/14] 00:25:55, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1404
192.168.0.6/32     *[L-ISIS/14] 00:25:55, metric 3000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1406
192.168.0.99/32    *[L-ISIS/14] 00:25:55, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1499
```

Given that R8 and R9 are not SR speakers, you need a way to generate node-SIDs on their behalf. This is achieved by using a mapping server. If R8 and R9 were SR speakers, a node SID would be configured on them, as seen in Chapter 3. Here, instead, you configure that same information on the mapping server. The mapping server can be any SR speaker in the domain. To emphasize this, let's use R1 as the mapping server, as it is nowhere near R6, R8, or R9. In practice, more than one mapping server would be deployed for resilience, as long as the mapping information configured on them is consistent. Configure R1 as follows:

```
user@R1# set protocols isis source-packet-routing mapping-server mapserver1

user@R1# set routing-options source-packet-routing mapping-server-entry mapserver1 prefix-
segment 192.168.0.8/32 index 408

user@R1# set routing-options source-packet-routing mapping-server-entry mapserver1 prefix-
segment 192.168.0.9/32 index 409
```

This configuration will trigger R1 to advertise the corresponding SIDs in ISIS. Now, at the inet.3 table on R2, you can see entries are now present corresponding to the loopback addresses of R8 and R9:

```
user@R2> show route table inet.3

inet.3: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.1/32     *[L-ISIS/14] 00:50:32, metric 1000
                    > to 10.0.0.0 via ge-0/0/0.0
192.168.0.3/32     *[L-ISIS/14] 00:48:34, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1403
192.168.0.4/32     *[L-ISIS/14] 00:42:03, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1404
192.168.0.6/32     *[L-ISIS/14] 00:42:03, metric 3000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1406
192.168.0.8/32     *[L-ISIS/14] 00:01:33, metric 4000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1408
192.168.0.9/32     *[L-ISIS/14] 00:01:33, metric 5000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1409
192.168.0.99/32    *[L-ISIS/14] 00:42:03, metric 2000
                    > to 10.0.0.0 via ge-0/0/0.0, Push 1499
```

Drill down into the ISIS database to see the Node SID information for R8 and R9 that was originated by R1:

```
user@R2> show isis database R1.00-00 extensive

<snip>
Label binding: 192.168.0.8/32, Flags: 0x10(F:0,M:0,S:0,D:1,A:0), Range 1
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 408
Label binding: 192.168.0.9/32, Flags: 0x10(F:0,M:0,S:0,D:1,A:0), Range 1
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 409

<snip>
```

The final piece of the configuration needed to achieve SR to LDP forwarding is as follows, on R6:

```
user@R6# set protocols isis source-packet-routing ldp-stitching
```

This triggers R6 to install entries in the mpls.0 table, to stitch incoming SR labels corresponding to R8 and R9 to the corresponding LDP label values. R8 advertised label 3 to R9 (penultimate hop-popping), hence the "pop" action for label 1408. R8 advertised label 54 for the loopback address of R9, hence the incoming label value of 1409 is swapped for label value 54:

```
user@R6> show route table mpls.0
<snip>

1408               *[L-ISIS/14] 00:00:05, metric 1000
                    > to 10.0.0.17 via ge-0/0/1.0, Pop
1408(S=0)          *[L-ISIS/14] 00:00:05, metric 1000
                    > to 10.0.0.17 via ge-0/0/1.0, Pop
1409               *[L-ISIS/14] 00:00:05, metric 2000
                    > to 10.0.0.17 via ge-0/0/1.0, Swap 54
<snip>
```

Let's now verify forwarding between the LDP and SR domains. First do a traceroute from CE6 to CE3:

```
user@CE6> traceroute 172.16.3.3
traceroute to 172.16.3.3 (172.16.3.3), 30 hops max, 52 byte packets
 1  10.3.1.6 (10.3.1.6)  2.728 ms  1.410 ms  1.382 ms
 2  10.0.0.20 (10.0.0.20)  9.313 ms  6.198 ms  6.063 ms
     MPLS Label=57 CoS=0 TTL=1 S=0
     MPLS Label=16 CoS=0 TTL=1 S=1
 3  10.0.0.16 (10.0.0.16)  6.876 ms  9.867 ms  9.303 ms
     MPLS Label=29 CoS=0 TTL=1 S=0
     MPLS Label=16 CoS=0 TTL=2 S=1
 4  10.0.0.8 (10.0.0.8)  8.186 ms 10.0.0.24 (10.0.0.24)  7.198 ms 10.0.0.8 (10.0.0.8)  6.665 ms
     MPLS Label=1402 CoS=0 TTL=1 S=0
     MPLS Label=16 CoS=0 TTL=3 S=1
 5  10.0.0.22 (10.0.0.22)  6.577 ms  6.219 ms 10.0.0.2 (10.0.0.2)  8.717 ms
     MPLS Label=1402 CoS=0 TTL=1 S=0
     MPLS Label=16 CoS=0 TTL=4 S=1
 6  10.3.1.2 (10.3.1.2)  6.804 ms  9.001 ms  6.202 ms
 7  172.16.3.3 (172.16.3.3)  7.305 ms  7.512 ms  7.920 ms
```

As you can see from this output, for the first hop the top label of the packet is the LDP label value of 57 allocated by R9's LDP neighbor R8, for destination R2. R8 in turn swaps that label for the LDP label value 29 that R6 allocated for destination R2. R6 swaps that label for R2's node-SID of 1402.

Now do a traceroute from CE3 to CE6:

```
user@CE3> traceroute 172.16.3.6
traceroute to 172.16.3.6 (172.16.3.6), 30 hops max, 52 byte packets
 1  10.3.1.2 (10.3.1.2)  2.131 ms  1.916 ms  6.172 ms
 2  10.0.0.0 (10.0.0.0)  7.810 ms  7.220 ms  7.255 ms
     MPLS Label=1409 CoS=0 TTL=1 S=0
     MPLS Label=22 CoS=0 TTL=1 S=1
 3  10.0.0.3 (10.0.0.3)  37.288 ms 10.0.0.23 (10.0.0.23)  84.775 ms 10.0.0.3 (10.0.0.3)  61.812 ms
     MPLS Label=1409 CoS=0 TTL=1 S=0
     MPLS Label=22 CoS=0 TTL=2 S=1
 4  10.0.0.25 (10.0.0.25)  7.778 ms 10.0.0.11 (10.0.0.11)  7.061 ms 10.0.0.25 (10.0.0.25)  11.256 ms
     MPLS Label=1409 CoS=0 TTL=1 S=0
     MPLS Label=22 CoS=0 TTL=3 S=1
 5  10.0.0.17 (10.0.0.17)  18.012 ms  7.241 ms  6.896 ms
     MPLS Label=54 CoS=0 TTL=1 S=0
     MPLS Label=22 CoS=0 TTL=4 S=1
 6  10.3.1.251 (10.3.1.251)  8.330 ms  6.909 ms  7.528 ms
 7  172.16.3.6 (172.16.3.6)  10.649 ms  7.466 ms  7.069 ms
```

As you can see, within the SR part of the network, the packet has the top label 1409 (the node-SID of R9) until it reaches R6. R6 swaps 1409 for the LDP label value of 54 that had been advertised by R8 to R6.

Summary

In this chapter, you have seen how interworking between shortest-path routing using Node Segments and LDP is achieved. This is a convenient way of dealing with situations in which some of the nodes in the network support LDP but not SR.

Chapter 5

Topology Independent Loop Free Alternate (TI-LFA)

This chapter shows you how the fast re-route mechanisms work in SPRING networks, as well as how to configure these mechanisms. The chapter assumes that you have a basic understanding of SPRING technology and that you have read the previous chapters in this book and now understand the different types of SIDs commonly used in SPRING networks.

General Introduction to Protection Mechanisms

Before starting a detailed discussion about protection and traffic restoration techniques, let's clarify some of the terminology used in this chapter.

When it comes to preparing the IP/MPLS network for protection of the traffic during various network failures, there are two major approaches:

- Protection based on *global repair*
- Protection based on *local repair*

These two approaches offer conceptual differences on how the network failure problem should be addressed and how the network should be prepared to handle such failures. They differ in the toolset used to handle traffic redirection during network failures and what failover times can be achieved. Having said that, in most of today's networks both methods are used in parallel: in other words, they complement each other.

Global Repair Concepts

During network failure events, the following actions lead to traffic redirection over a new path to avoid a failed link or node:

1. Local failure detection:

- The time period required to locally detect the failure.
- Various detection techniques are available (for example, loss of signal, CFM/BFD failures, etc.), depending on the underlying physical transport technology.
- It can take from micro- or milliseconds for the loss of signal, up to seconds (for example, failure detection based on loss of IGP Hello messages).

2. New state propagations (flooding):

- The time period required to propagate the information about the failed link or node through the network.
- Typically involves IGP (IS-IS or OSPF) flooding.
- This time depends greatly on the size of the network, link distances, and other factors unique to your environment. For example, it takes approximately 5ms. to send a signal across 1000 km. Additionally, each transit IGP router might add some time (typically 10-100 ms.), as it might wait to receive more updates so that all multiple updates can be sent at once.

3. Routing database updates and new path (and label) computations:

- The time required to compute new paths (next hops).
- Depends on the IGP database size. On modern, high-end routers, this can be approximated with around 10 µs per node (in a network with 1,000 nodes it then takes approximately 10 ms to perform a full shortest-path first [SPF] calculation).

4. Establish new next hops (and labels) installation in the hardware forwarding information base (HW FIB):

- The time required to program HW FIB in the line cards with newly calculated next hops (labels).
- It's very hardware dependent.
- It can take a relatively long time (measured in seconds) for a large number of next hops in a scaled environment.

By tuning IGP parameters (for example SPF timers, and flooding timers, defining prioritization among prefixes to first program HW FIB with next hops related to loopbacks, etc.), you can achieve sub-second global convergence in most modern network deployments, and up to a few seconds in very large networks with old routing engines (therefore, slow CPUs, long time to program HW FIB). However, achieving sub-100ms convergence, global (network-wide) convergence is difficult, because the state propagation, routing database update, new path calculation, and installation of new next hops in HW FIB cannot really be squeezed below a couple of 100ms. Thus, for very demanding applications that require sub-100ms traffic failover times during network failures, tuning global convergence parameters alone is no longer enough. In these cases, *local repair* comes into the picture.

Local Repair Concepts

The idea behind local repair is to skip most of the steps that must happen with global repair when a network failure happens. If, on the node detecting local failure (when discussing local repair techniques such nodes are commonly called *Point of Local Repair – PLR*), another (backup) next hop redirecting the traffic around the failure was already installed in HW FIB, the only action that needs to be performed during failure events is to detect the failure itself and remove the original (primary) next hops associated with the failed link or node from the HW FIB. All the other steps are no longer required for local repair. With such an approach, the times it takes to repair the traffic (redirect the traffic around network failure) can be squeezed from subsecond/second (using global convergence mechanisms) to sub-100ms, or even sub-50ms, in many cases.

Strictly speaking, local repair is complementary (and not an alternative) to global repair. Indeed, local repair and global repair take place in parallel. Local repair quickly reroutes the traffic around the failure by using a pre-computed and pre-installed temporary backup path while global repair computes the final converged path.

While preparation of the network for global convergence is typically straightforward (basically tuning some parameters), local repair is much more challenging. The backup next hop must be selected and programmed at least in a way to avoid traffic looping. Local repair happens very fast. Often traffic is redirected over a backup next hop within 50ms (the term frequently used to describe local repair is fast reroute), therefore traffic redirected by PLR traverses nodes that have an old (pre-failure) view of the network, since global convergence still didn't happen on them. Therefore, these nodes may still believe the shortest path to the destination is via failed link or node. Thus, the challenge is how to ensure that in such transient state loop doesn't occur, and that will be discussed in subsequent sections of this chapter.

SPRING Fast Traffic Restoration Basic Concepts

The basic idea with SPRING protection is very simple: instead of redirecting the traffic around failure over some temporary path, and then, after global convergence happens, redirecting that same traffic again using a newly calculated global path, you make the backup path, right from the beginning, equal to the shortest post-convergence path. This idea, enhanced with additional details of course, is described in *draft-bashandy-rtgwg-segment-routing-ti-lfa* (*Topology Independent Fast Reroute using Segment Routing*).

Is this idea new and innovative? Certainly not. This idea has already been in use in IP/MPLS networks for many years. Namely, RFC 4090 (*Fast Reroute Extensions to RSVP-TE for LSP Tunnels*), published in 2005, introduced a technique called *One-to-One Backup* (Section 3.1 of RFC 4090). It is nothing other than but using post-convergence SPF paths for backup purposes! The main difference between RSVP 1:1 backup and SPRING fast-reroute is how the backup path is set up.. RSVP uses signaling to allocate labels and create states across all transit nodes along the backup path from PLR to the final destination, while in SPRING the backup path is enforced by using appropriate label stack pushed by PLR on the top of the redirected packet. In both cases (RSVP 1:1 and SPRING) the shortest post-convergence backup path can be created (enforced) in any arbitrary topology – it is topology independent. Therefore, in SPRING terminology, this kind of protection is called *Topology Independent Loop Free Alternate* (TI-LFA).

How can the post-convergence path be calculated? Very simply: just use the for the SPF calculation modified link state database (LSDB), where the network resource (for example, link or node) being protected is removed from the LSDB.

In general, when discussing fast re-route in SPRING networks, the following terms are used:

Table 5.1 TI-LFA Terminology

LSDB_old	LSDB in the initial state of the network (before failure).
LSDB_new(X)	LSDB after failure (in the state after resource X has failed).
SPT_old(R)	The shortest-path tree (SPT) rooted at node R in the initial state of the network (before failure).
SPT_new(R, X)	The SPT rooted at node R in the state of the network after the resource X has failed.
Dist_old(A,B)	The distance (IGP path metric) from any node A to any node B in SPT_old(A) – before failure.
Dist_new(A,B, X)	The distance (IGP path metric) from any node A to any node B in SPT_new(A,X) – after the resource X has failed.

Backup Next-Hop Selection

Let's start the discussion by understanding the difference in backup next-hop selection between legacy LFA and TI-LFA, using modified (adjusted IGP metrics) network topology, as illustrated in Figure 5.1.

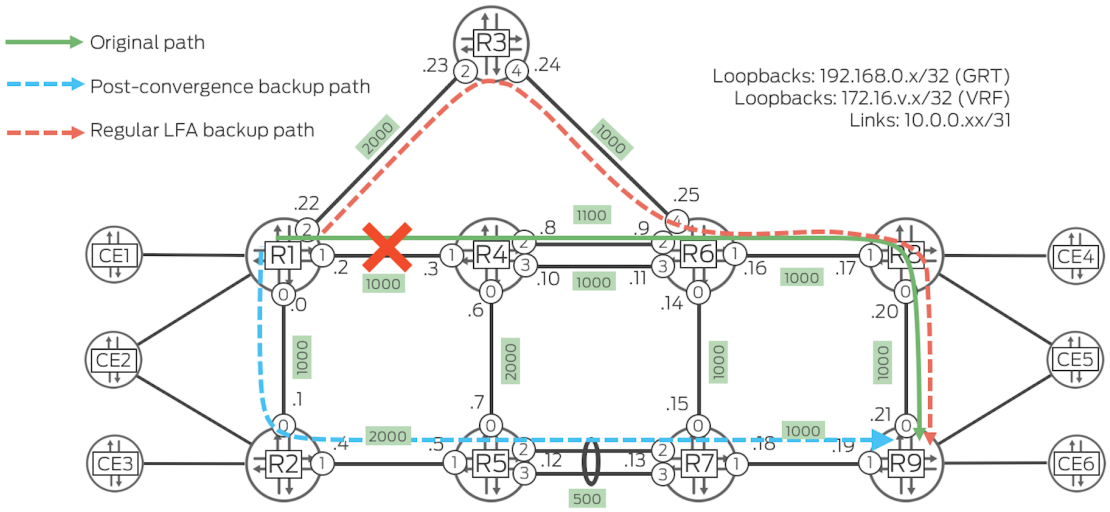

Figure 5.1 *TI-LFA Network Topology*

In Junos implementation, legacy LFA selects the backup next hop closest to the destination (please note, RFC 5286 – *Basic Specification for IP Fast Reroute: Loop-Free Alternates* – does not specify which LFA backup next hop should be selected when there are multiple non-looping backup next hops possible; in such case, it is up to the vendor implementation to freely chose the backup next hop). In this particular case, to reach R9 from R1, the shortest path is via R4, with total path cost 4000. The legacy LFA backup next hop is R3, with path cost from R3 to R9 equal to 3000. Legacy LFA does not select R2 as the backup next hop, since the path cost from R2 to R9 is higher (3500) than from R3 to R9 (3000). TI-LFA, on the other hand, takes into account total post-convergence cost (from the source R1) of the backup path. Via R3 it is 5000, while via R2 it's only 4500. Therefore, the choice for TI-LFA backup next hop is R2.

Let's verify these assumptions in the real network topology. The legacy (R)LFA configuration, using SPRING (not LDP) as the tunneling technology for RLFA, is:

```
groups {
    GR-ISIS {
        protocols isis interface <*e*> {
            node-link-protection;                # Prefer node protection backup next-hop
        }
    }
}
protocols {
    isis {
        apply-groups GR-ISIS;
        backup-spf-options {
            remote-backup-calculation;           # Use RLFA next-hop, when necessary
            node-link-degradation;             . # Use link protection next-hop as last resort
            use-source-packet-routing;           # Protect both, MPLS and native IP traffic
        }
    }
}
```

NOTE The `use-source-packet-routing` configuration knob enables SPRING-based LFA protection for prefixes in inet.0 (plain IPv4 traffic) and inet6.0 (plain IPv6 traffic) RIBs. Without this knob being configured, only SPRING prefixes in inet.3 (ingress labeled IPv4 traffic), inet6.3 (ingress labeled IPv6 traffic), and mpls.0 (transit labeled traffic) RIBs are subject for SPRING-based LFA protection.

The chosen backup next hop is R3, which confirms our earlier discussion:

```
kszarkowicz@R1# run show route 192.168.0.9 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0x1, selected   # R4
                Label operation: Push 1409
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0xf000         # R3
                Label operation: Push 1409

kszarkowicz@R1# run show isis backup spf results level 2 R9
 IS-IS level 2 SPF results:
R9.00
  Primary next-hop: ge-0/0/1.0, IPV4, R4, SNPA:  56:68:a3:1a:19:ac
    Root: R9, Root Metric: 4000, Metric: 0, Root Preference: 0x0
      track-item: R9.00-00
      Not eligible, IPV4, Reason: Missing primary next-hop
      Not eligible, LSP, Reason: Interface is already covered
    Root: R7, Root Metric: 3000, Metric: 1000, Root Preference: 0x0
      track-item: R7.00-00
      Not eligible, IPV4, Reason: Missing primary next-hop
      Not eligible, LSP, Reason: Interface is already covered
    Root: R8, Root Metric: 3000, Metric: 1000, Root Preference: 0x0
      track-item: R8.00-00
      Not eligible, IPV4, Reason: Missing primary next-hop
      Not eligible, LSP, Reason: Interface is already covered
    Root: R5, Root Metric: 3000, Metric: 1500, Root Preference: 0x0
      track-item: R5.00-00
      Not eligible, IPV4, Reason: Missing primary next-hop
      Not eligible, LSP, Reason: Interface is already covered
    Root: R6, Root Metric: 2000, Metric: 2000, Root Preference: 0x0
```

```
      track-item: R6.00-00
      Not eligible, IPV4, Reason: Missing primary next-hop
      Not eligible, LSP, Reason: Interface is already covered
    Root: R4, Root Metric: 1000, Metric: 3000, Root Preference: 0x0
      Not eligible, IPV4, Reason: Primary next-hop link fate sharing
    Root: R3, Root Metric: 2000, Metric: 3000, Root Preference: 0x0
      Eligible, Backup next-hop: ge-0/0/2.0, IPV4, R3,  SNPA:  56:68:a3:1a:19:b7,Prefixes: 1
    Root: R2, Root Metric: 1000, Metric: 3500, Root Preference: 0x0
      Not eligible, IPV4, Reason: Interface is already covered
  1 nodes
```

Now, let's replace the legacy (R)LFA configuration (using SPRING and not LDP as the tunneling technology) with a TI-LFA (native SPRING fast reroute architecture) configuration, as outlined in following example:

```
groups {
    GR-ISIS {
        protocols isis interface <*e*> {
            level 2 post-convergence-lfa;
        }
    }
}
protocols {
    isis {
        apply-groups GR-ISIS;
        backup-spf-options {
            use-post-convergence-lfa;
            use-source-packet-routing;
        }
    }
}
```

This time, the chosen backup next hop is R2:

```
kszarkowicz@R1# run show route 192.168.0.9 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0x1, selected   # R4
                Label operation: Push 1409
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000          # R2
                Label operation: Push 1409
```

There are couple of operational commands that allow you to verify TI-LFA status:

```
kszarkowicz@R1# run show isis overview | match Backup
  Post Convergence Backup: Enabled
    Max labels: 3, Max spf: 100, Max Ecmp Backup: 1


kszarkowicz@R1# run show isis interface ge-* extensive | match "ge-|Protection"
ge-0/0/0.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 0
ge-0/0/1.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 0
ge-0/0/2.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 0
```

The exact meaning of the information presented in these operational command outputs will be discussed later in this chapter.

NOTE The `show isis backup` operational command is not meaningful with TI-LFA, since TI-LFA always – irrespective of the network topology as the name suggests *Topology Independent* LFA – produces 100% backup coverage and produces some backup next hops for each prefix.

Link Versus Node Protection

TI-LFA, by default, uses link-protection backup next hops. This can be verified, for example, on router R4:

```
kszarkowicz@R4# run show route 192.168.0.9 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.11 via ge-0/0/3.0 weight 0x1, selected   # R6
                Label operation: Push 1409
                Next hop: 10.0.0.9 via ge-0/0/2.0 weight 0xf000           # R6
                Label operation: Push 1409
```

You can see that both the primary and backup next hops are R6 – just using different links. Therefore, when the primary (bottom) R4-R6 link fails, the second (upper) R4-R6 link is used as backup. However, what happens when node R6 itself fails? Bad luck! Your traffic will not be protected, despite the fact that another backup option (via R5) exists. R5 is not used, since R5 is not on the shortest post-convergence path for protection of the bottom R4-R6 link.

To use node protection with TI-LFA, you must enable it on the IGP interface level:

```
groups {
    GR-ISIS {
        protocols isis interface <*e*> {
            level 2 post-convergence-lfa node-protection;
        }
    }
}
protocols {
    isis {
        apply-groups GR-ISIS;
    }
}
```

This time the backup next-hop is R5, therefore traffic is protected not only against the failure of the link from R4 to R6, but also against the failure of R6 node as well:

```
kszarkowicz@R4# run show route 192.168.0.9 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
```

```
Next hop: 10.0.0.11 via ge-0/0/3.0 weight 0x1, selected   # R6
Label operation: Push 1409
Next hop: 10.0.0.7 via ge-0/0/0.0 weight 0xf000           # R5
Label operation: Push 1409
```

But why now (when `node-protection` is enabled) is R5 on the shortest post-convergence path, while before (without enabling `node-protection`) it was not? It comes back to the way which kind of post-convergence LSDB is used: LSDBnew(X). IIn case of link protection (the default TI-LFA behavior), the 'X' resource is the link, so the next hop is determined by running SPF on LSDB with the link removed. When node protection is used, though, LSDB is modified by temporarily removing the R6 node. In such a modified LSDB, R5 is now the next hop on shortest post-convergence path.

There is one more case that requires special attention. Let's assume a modified network topology, where router R7 doesn't exist, as shown in Figure 5.2.

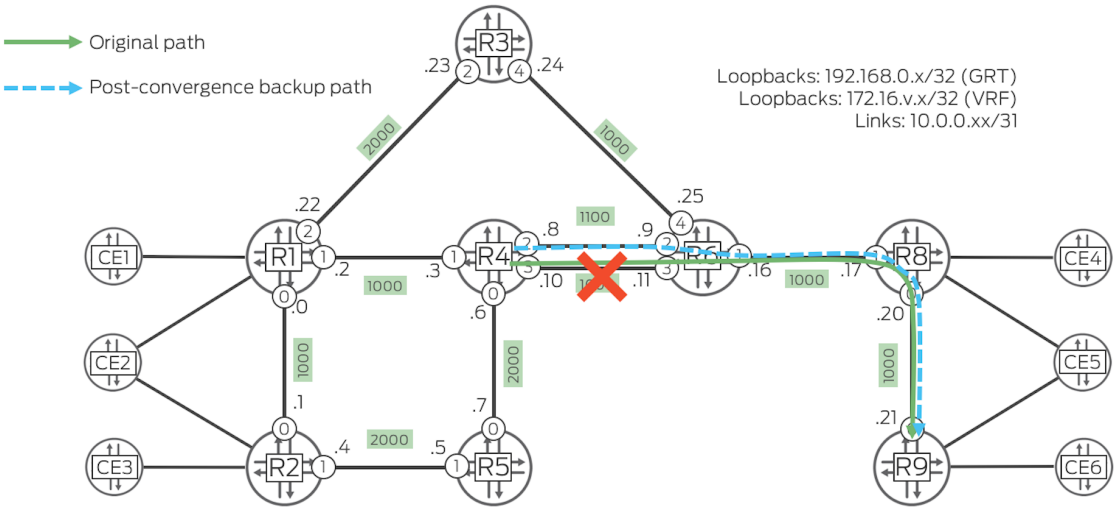

Figure 5.2 *TI-LFA Network Topology (Node Protection with Link Protection Fallback)*

In Figure 5.2's topology, when TI-LFA node protection is enabled, there is no backup next hop, because going through node R6 is the only way to reach R9:

```
kszarkowicz@R4# run show route 192.168.0.9 table inet.3 detail |
              match "entry|L-ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.11 via ge-0/0/3.0 weight 0x1, selected   # R6
                Label operation: Push 1409
```

Therefore, it would be beneficial to fall back to link protection, where node protection is not possible. In legacy (R)LFA configurations, you could use the `node-link-degradation` knob to enable such behavior.

TI-LFA uses a different approach, namely you can use *strict* or *loose* node protection. With *strict* node protection mode, as already discussed, the protected node is completely removed from LSDB during SPF calculations to determine the backup next hop. With *loose* node protection mode, instead of completely removing the protected node from the LSDB, the cost of all the links (other than the primary link, for which the backup path is being calculated) to the protected node will be increased by the configured node protection cost value. Thus, the use of the protected node will be strongly (if configured node protection cost value is high) discouraged, but the protected node will be still considered as a backup, even if there is no other option. Therefore, it can be used as a fallback to link protection, if node protection is not possible.

To enable this behavior, you need to configure the link cost (link metric) for node protection:

```
groups {
    GR-ISIS {
        protocols isis interface <*e*> {
            level 2 post-convergence-lfa node-protection cost 16777214;
        }
    }
}
protocols {
    isis {
        apply-groups GR-ISIS;
    }
}
```

CAUTION Configuring `node-protection cost` to be equal to the maximum IGP link metric (IS-IS: 16777215, OSPF: 65535) keeps the strict node protection behavior in place. You need to configure `node-protection cost` to be smaller (at least by 1) than the maximum IGP link metric to enable loose node-protection behavior.

With the changes being made you can now see new node cost being used:

```
kszarkowicz@R4# run show isis interface ge-* extensive | match "ge-|Protection"
ge-0/0/0.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 16777214
ge-0/0/1.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 16777214
ge-0/0/2.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 16777214
ge-0/0/3.0
    Post convergence Protection:Enabled, Fate sharing: No, Srlg: No, Node cost: 16777214
```

...and the second R6 link as the backup next hop:

```
kszarkowicz@R4# run show route 192.168.0.9 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.9/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.11 via ge-0/0/3.0 weight 0x1, selected   # R6
                Label operation: Push 1409
                Next hop: 10.0.0.9 via ge-0/0/2.0 weight 0xf000           # R6
                Label operation: Push 1409
```

ECMP Backup Next Hops

Now let's discuss the problem using a slightly modified topology as illustrated in Figure 5.3.

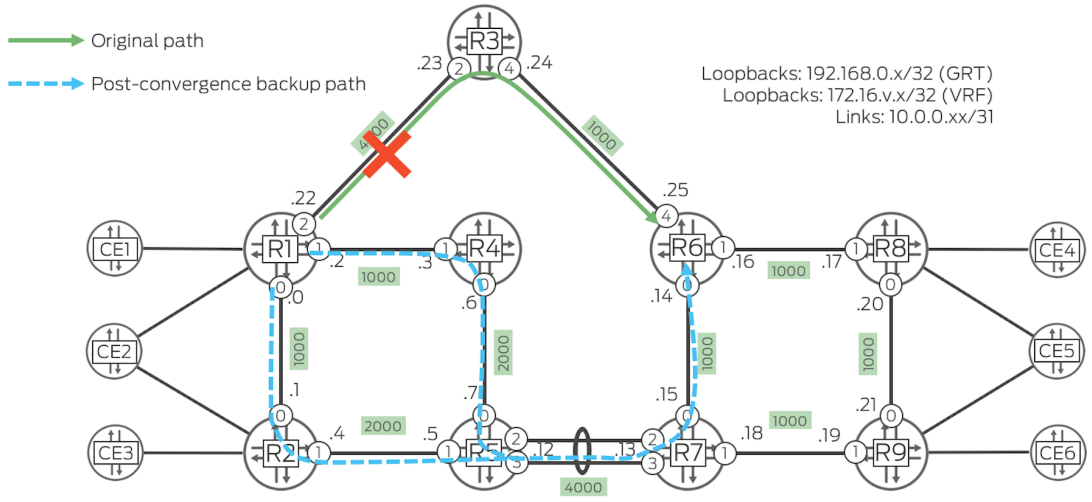

Figure 5.3 *TI-LFA Network Topology (ECMP Backup Next Hops)*

From R1 you can reach R6 over R3, as the primary next hop with total path cost 5000, and via R2/R4, as ECMP backup next hops with a total backup path cost 8000. Verifying the state of the RIB, however, outlines that only one out of two possible ECMP backup next hops are used:

```
kszarkowicz@R1# run show route 192.168.0.6 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.6/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Push 1406
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000          # R2
                Label operation: Push 1406, Push 1407(top)
```

You might recall the output of the `show isis overview` command discussed earlier, showing maximum ECMP backup next hops equal to 1. To enable more than one ECMP backup next hop, you must configure the number explicitly:

```
protocols {
    isis {
        backup-spf-options {
            use-post-convergence-lfa maximum-backup-paths 2;
        }
    }
}
```

Verification confirms, two backup next hops are now taken into account, therefore when the link towards R3 fails, traffic will be load-balanced over two next hops (R2 and R4):

```
kszarkowicz@R1# run show isis overview | match Backup
  Post Convergence Backup: Enabled
    Max labels: 3, Max spf: 100, Max Ecmp Backup: 2

kszarkowicz@R1# run show route 192.168.0.6 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.6/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Push 1406
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000           # R2
                Label operation: Push 1406, Push 1407(top)
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0xf000           # R4
                Label operation: Push 1406, Push 1407(top)
```

NOTE If more than the configured number of backup next hops are found, unnecessary backup next hops are pruned. The pruning is deterministic with the pruning algorithm being proprietary to Junos implementation.

Fate-Sharing

For a backup path to work effectively it must not share links or physical fiber paths with the primary path, ensuring that a single point of failure will not affect the primary and backup paths simultaneously.

Fate-sharing allows extension of a local database (not distributed between routers), that TI-LFA uses to compute backup paths. The fate-sharing entries describe the relationships between elements of the network, such as routers and links. Within a fate-sharing group, you should specify at least two entries (for example, two links), to indicate they are sharing a fate.

Each fate-sharing group, apart from being associated with at least two network elements (links) sharing the fate, is as well associated with some configurable cost (default is 1, if not configured explicitly). Similar to loose node protection, when TI-LFA computes backup paths, cost of all network elements (links) sharing fate with network elements of the primary path will be increased by this value.

Thus, the use of network elements sharing fate will be highly discouraged (if the configured fate-sharing cost value is high), but these network elements will be still considered as backup, if there is no other option. Therefore, through fate-sharing, you can configure backup paths that minimize, as much as possible, the number of shared links and fiber paths with the primary paths. That ensures that if a fiber is cut, a minimum amount of data is lost and a path still exists to the destination.

To verify fate-sharing operation, let's assume (using the network topology from Figure 5.3) that R1-R3 and R4-R5 links are (partially) carried over the same fiber infrastructure. Therefore, the backup path should avoid R4-R5 link. Let's consider this configuration on R1:

```
groups {
    GR-ISIS {
        protocols isis interface <*e*> {
            level 2 post-convergence-lfa fate-sharing-protection;
        }
    }
}
routing-options {
    fate-sharing {
        group FS-TEST {

cost 65535;                  # cost added to fate sharing links during backup path calculation
            use-for-post-convergence-lfa; # use this fate sharing group during TI-LFA calculations
            from {
                10.0.0.6 to 10.0.0.7;     # R1-R3 link
                10.0.0.22 to 10.0.0.23;   # R4-R5 link
            }
        }
    }
}
protocols {
    isis {
        apply-groups GR-ISIS;
    }
}
```

NOTE If the `cost` configuration knob is omitted, the default value used during TI-LFA calculations is 1.

Apart from defining standard fate-sharing groups (similar to fate-sharing groups used in RSVP) you must designate the fate-sharing group as well for usage during TI-LFA calculations (normally it can be used only during RSVP calculations), as well as enable TI-LFA fate-sharing protection on the interface level. Verification confirms, that TI-LFA fate-sharing is now enabled:

```
kszarkowicz@R1# run show mpls fate-sharing
Group FS-TEST {
  Cost 65535
  10.0.0.6  10.0.0.7
  10.0.0.22  10.0.0.23
}
```

```
kszarkowicz@R1# run show isis interface ge-* extensive | match "ge-|Protection"
ge-0/0/0.0
    Post convergence Protection:Enabled, Fate sharing: Yes, Srlg: No, Node cost: 16777214
ge-0/0/1.0
    Post convergence Protection:Enabled, Fate sharing: Yes, Srlg: No, Node cost: 16777214
ge-0/0/2.0
    Post convergence Protection:Enabled, Fate sharing: Yes, Srlg: No, Node cost: 16777214
```

WARNING The `show mpls fate-sharing` operational command is a hidden command. You need to type the command exactly, since there is no auto-completion for hidden commands.

And the backup next hop R4 is no longer used, due to the fact that the R4-R5 link shares fate with the R1-R3 link from the primary path:

```
kszarkowicz@R1# run show route 192.168.0.3 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.3/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected   # R3
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000           # R2
                Label operation: Push 1403, Push 1406(top)
```

NOTE TI-LFA SRLG protection mentioned in *draft-bashandy-rtgwg-segment-routing-ti-lfa* is not qualified in Junos 17.4 release.

Label Stack for TI-LFA Backup Path

You might realize in the examples shown so far that sometimes the backup next hop uses a single label, but in some other case, there is a label stack of two labels associated with the backup next hop. Thus, a legitimate question can be raised: *How is the label stack for backup path actually determined?*

The most straightforward answer would be: Use an Adj-SID for each link (each IGP adjacency) along the backup path to enforce strict path routing through the network. With this method the required label stack can very easily be determined: you simply take the result of SPT_new(R, X) calculation, determine all the links (IGP adjacencies) provided by this calculation, and use the Adj-SID associated with each link (IGP adjacency) to create a label stack that should be pushed on the packet forwarded over the backup next hop.

Is this approach clever enough to be actually implemented in the real network? Probably not! The biggest problem with this approach is that it can potentially produce very large label stacks (containing large numbers of Adj-SIDs) that should be pushed by the PLR on the packet. Why is it a problem? There are couple of issues:

- The PLR might not be able to push large label stacks on the packet. Typical legacy hardware can push a maximum 3-5 labels, while 'better' hardware, such as the Trio chipset, can push more (for example, 16).
- Transit routers on the backup path might not be able to hash through the entire (large) label stack and IP header beyond that stack for load-balancing purposes. Therefore, when the load-balancing need arises somewhere on transit (for example, load-balancing over member links of the Ethernet LAG bundle interface), it might be problematic.
- Packet size might increase significantly. For large packets, pushing additional large headers might have implications for the MTU (Maximum Transmit Unit). It might happen that the packet size with the large header is bigger than the actual MTU size, causing problems with packet transmission (or, requiring fragmentation, which is never good, and not even supported in newer protocols such as IPv6). For small packets, the implications could be bandwidth provisioning over backup path. For example, if you have 1 Gbps worth of traffic (predominantly using small packets, say VoIP) on the primary path, and because of some network failure this traffic must be redirected over the backup path, it can suddenly become 2 Gbps, due to the large header (in relation to the packet size itself) imposed on the small packets on the backup path. You can easily imagine small VoIP packets (G.729 codec, typically used in mobile networks, uses packets with only 20 bytes payload) with a large MPLS header (a MPLS header of 10 labels takes 40 bytes) on the backup path.

Therefore, the conclusion is that large (MPLS) headers are never good and should be avoided, if possible. It's the golden rule in any decent network design.

Now, let's go back to SPRING and TI-LFA. Is this simple approach (Adj-SID for each link on the backup path) actually used? Fortunately – not! If you review all previously discussed examples, you can find (look at the diagrams) backup paths going through 3, 4, or 5 routers. However, the label stacks associated with backup next hops in these examples only have 1 or 2 labels – certainly, the label stack size doesn't reflect the length of the backup path. Uff!

If you think over the entire problem of TI-LFA backup paths, what is really needed – from the perspective of MPLS headers – is the label stack, which is sufficient to forward the packet from PLR to final destination over shortest post-convergence path (calculated based on post-convergence LSDB_new(X)), keeping in mind that the transit routers on the backup path will use the result of the SPF calculation based on the pre-convergence database (LSDB_old). In other words, switching the traffic to the backup next hop (around ~50 ms) is much faster than the IGP convergence (a few hundred ms up to a few seconds), so all transit routers will forward the traffic based on pre-failure LSDB (LSDB_old). So, the imposed label stack, while is not required to be an explicit hop-by-hop (no need for Adj-SID of each link) must achieve loop-free forwarding over the backup path. Here comes the LFA (loop-free alternate) part of the TI-LFA acronym.

The actual procedure to determine the list of segments with minimum length, but which still fulfills previously described loop free requirements is essentially vendor proprietary. Each vendor might have their own way to determine it. Regardless, this doesn't require standardization, since if the resulting label stack has two, or three, or even four labels, if the SPRING label stack semantic is fulfilled, the packet can be successfully forwarded across the SPRING network.

That being said, *draft-bashandy-rtgwg-segment-routing-ti-lfa* provides some hints how it can be done. To determine the list of segments (label stack) pushed on the packet sent over backup path, *draft-bashandy-rtgwg-segment-routing-ti-lfa* uses the following concepts:

- P-Space: This is a set of routers reachable from a PLR router (denoted *S*) using a shortest path (using LSDB_old) and without traversing the protected link. In the case of ECMP, this requirement applies to all the equal-cost shortest paths from *S* to a node in the P-space. None of these paths can traverse the protected link; otherwise, the node is not in the P-space.
- Extended P-space: The union of P-space computed for PLR router (denoted *S*) as well as P-spaces computed for each direct neighbor of *S*, excluding primary next-hop router (denoted *E*).
- Q-Space: This is a set of routers that can reach the primary next hop (denoted *E*) using a shortest path (using LSDB_old) and without traversing the protected link. In the case of ECMP, this requirement applies to all the equal-cost shortest paths from a node in the Q-space to *E*.
- PQ-node: This is a node that is a member of both the extended P-space and the Q-space. The PQ-node does not need to be directly connected to *S* (or to *E*).

To determine PQ-nodes, PLR performs the following multiple SPF calculations (using LSDB_old):

- One *primary* SPF calculation, using itself (PLR) as the root of the SPF tree. Routers always perform these types of SPF calculations, regardless of whether LFA is enabled, to determine primary next hops due to normal IGP operation.
- Multiple *backup* SPF calculations, with each calculation using a different direct IGP neighbor node as the root of the SPF tree. Routers perform this type of SPF calculation to only determine backup next hops, if the LFA feature is enabled.

No Additional Label for TI-LFA Backup Path

Figure 5.4 shows an example of P-space and Extended P-space in the network topology used at the beginning of this chapter.

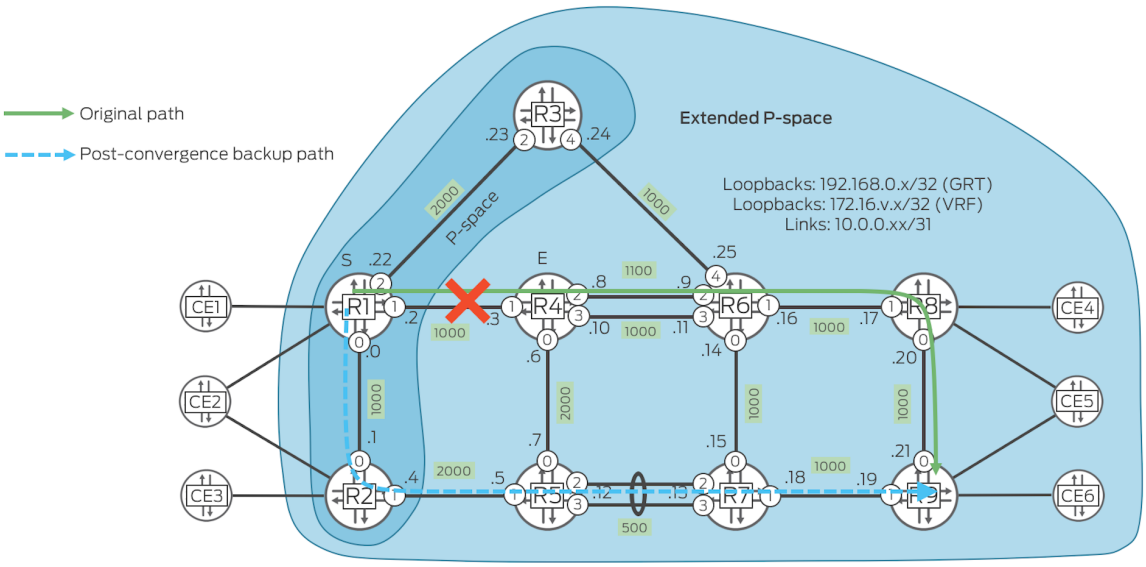

Figure 5.4 *TI-LFA P-Space and Extended P-Space*

As you can see, router R1 can reach only routers R2 and R3 without crossing R1>R4 link (link under protection). It is the P-space of R1 for the R1>R4 link. On the other hand, you can reach the rest of the topology without crossing link under protection either from R2 or R3 – direct neighbors of R1. This is the Extended P-space of R1 for R1>R4 link. If the TI-LFA (post-convergence) backup path towards the destination (in this case R9) traverses only nodes found in the Extended P-space, the calculation of Q-space is not required. Therefore, in this example, R1 performed only three SPF calculations, each time placing different routers (R1, R2, R3) as the root of the SPF tree.

NOTE If the TI-LFA (post-convergence) backup path uses only nodes in the P-space, single SPF calculation is enough – there is no need to discover Extended P-space.

If the TI-LFA backup path towards its destination uses only the routers from P-space or Extended P-space (including the destination router), when traffic is being redirected over the backup path during failure event, then no additional labels must be pushed on the redirected packet (verify this with the `show` outputs already presented earlier when this topology was first discussed).

As per (Extended) P-space definition, traffic from such node can reach other nodes in (Extended) P-space (including destination node) without crossing the link under protection – and still using the old (pre-failure) IGP database. So, no special attention is required regarding loop prevention. In other words, you don't need the list of explicit next hops for TI-LFA backup path. You can simply redirect the traffic!

Single Additional Label for TI-LFA Backup Path

When you look at the second example, exploiting the etwork topology already used during discussion of ECMP backup next hops (Figure 5.5), you realize that the destination node (R6) doesn't belong to either P-space or to Extended P-space.

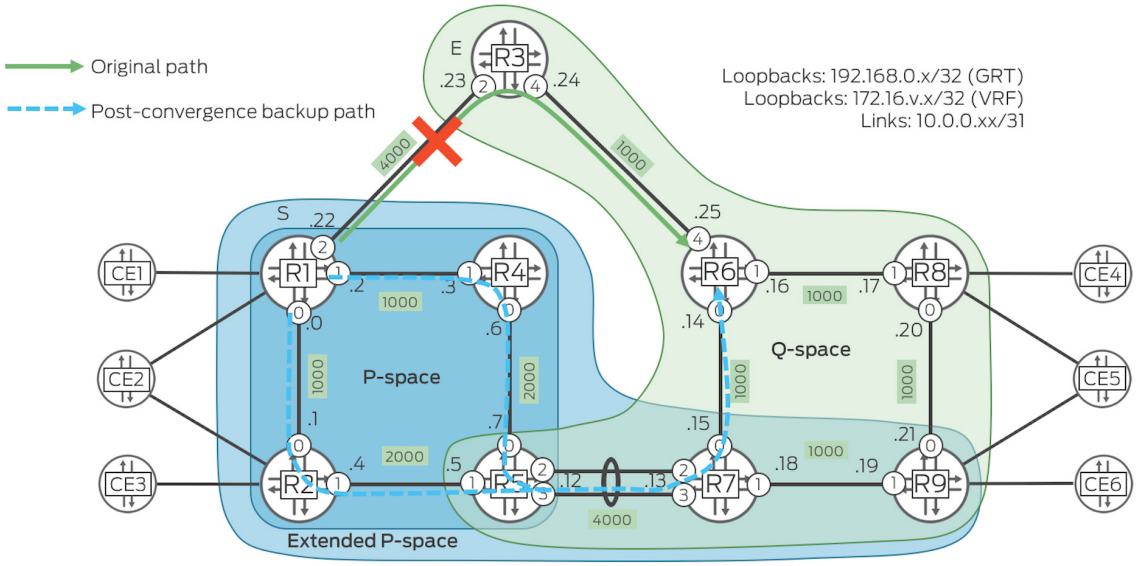

Figure 5.5 *TI-LFA (Extended) P-space and Q-space*

That is, neither from R1, nor from R1's direct neighbors (R2 or R4) can you reach R6 without crossing the R1>R3 link. Therefore, simple redirection, as used before, would not help, since the traffic would be looped back towards R1.

The solution of the problem is to find routers on the TI-LFA post-convergence backup paths that belong to both (Extended) P-space and Q-space. Such routers are called *PQ-nodes*. In our example, PQ-nodes are the routers R5, R7, and R9. It is enough to force the traffic to some PQ-node, since from PQ-node, based on Q-space definition, the shortest path towards the destination doesn't use link under protection.

But an important question is how the Q-space is determined from a computational perspective? Calculating SPF routed at each router in the network to verify that R3 can be reached without crossing R1>R3 link would be very computationally extensive. Imaging you are managing a large network of several thousand routers – R1 router would need to run several thousand SPF calculations.

That doesn't scale well. However, what you can do quite easily is to use a reverse SPF calculation putting R3 at the root of the SPF tree. Reverse here means that link metrics towards (not onwards) R3 are using during SPF calculation. With this little trick, it is enough to make a single SPF calculation to determine the Q-space.

NOTE In Q-space calculations, only the primary next hop node is taken into account, not the actual destination node. Calculating the Q-space for every destination node would, in the worst case, require a reverse SPF computation rooted on many nodes for each destination, which would be non-scalable in large networks. Therefore, the Q-space of E is used as a proxy for the Q-space of each destination.

Let's review the status of the routing entry on R1:

```
kszarkowicz@R1# run show route 192.168.0.6 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.6/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Push 1406
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000           # R2
                Label operation: Push 1406, Push 1407(top)
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0xf000           # R4
                Label operation: Push 1406, Push 1407(top)
```

You can see that primary next hop has a single label. Backup next hops (you remember, these are ECMP backup next hops) use two labels: 1406 (R6 Node-SID) and top label 1407 (R7 Node-SID). Each Node-SID is a representation of the loose next hop in SPRING LSP. So, basically, these two SIDs mean:

- Send the packet over the shortest path towards R7 (top label);
- Next, from R7, send the packet over the shortest path towards R6 (bottom label).

According to the definition of (Extended) P-space, R7 (belonging to Extended P-space) can be reached from R1 neighbor (for example, from R2 or R4) without crossing the R1>R3 link. Further, you can reach R6 from R7 without crossing the R1>R3 link, again, due to the fact that R7 belongs to Q-space, too.

Two Additional Labels for the TI-LFA Backup Path

The next example uses a topology diagrammed in Figure 5.6.

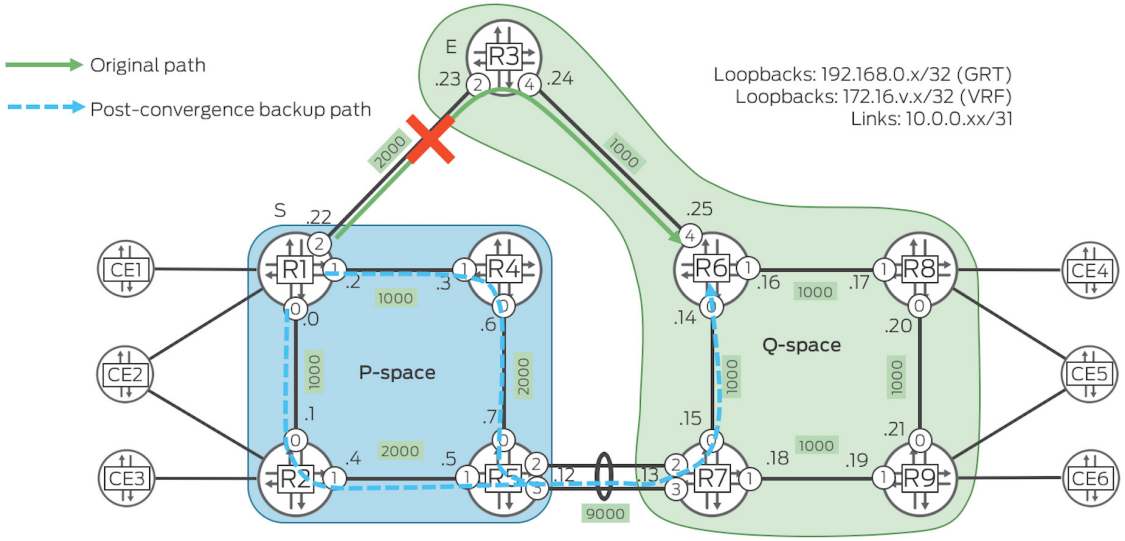

Figure 5.6 *TI-LFA with no overlap between P-Space and Q-Space*

This time, there is no overlap between (Extended) P-space (in the example, P-space is equal to Extended P-space) and Q-space. Consequently, there is no PQ-node. The way traffic can be forwarded in such a scenario is as follows:

- Send the packet over the shortest path towards the P-node, which has the Q-node as a direct neighbor. Such nodes in the topology are node R5 (P-node) and node R7 (Q-node).
- Next, from such node, use Q-node as a strict next hop. That is, send the traffic over a direct link towards Q-node.
- Next, from Q-node (R7), send the packet over shortest path towards destination.

Let's check the status of the routing entry on R1:

```
kszarkowicz@R1# run show route 192.168.0.6 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.6/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Push 1406
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000                 # R2
                Label operation: Push 1406, Push 50007, Push 1405(top)
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0xf000                 # R4
                Label operation: Push 1406, Push 50007, Push 1405(top)
```

As you can see, the top label of the backup next hops is the Node-SID of R5 (R5 as a loose next-hop). The next-label is the unprotected Adj-SID of the R5-R7 adjacency (R7 as a strict next hop), as visible in the following output:

```
kszarkowicz@R7# run show isis database R5 detail | find "IS neighbor: R7" | except "IP prefix"
IS neighbor: R7.00                          Metric:     9000
     P2P IPv4 Adj-SID:     507, Weight:   0, Flags: -BVL-P
     P2P IPv4 Adj-SID:   50007, Weight:   0, Flags: --VL-P
```

The last (bottom of stack) label is R6 Node-SID (R6 is a loose next hop). Therefore, it can be concluded that two additional labels were required to program the TI-LFA backup path in this use case.

In summary, three use cases for TI-LFA link-protection discussed so far are:

1. A TI-LFA backup path traverses over P-nodes only – no additional MPLS backup label is required for protection.
2. TI-LFA backup paths traverse over the PQ-node – one additional MPLS backup label (Node-SID associated with PQ-node) is required for protection.
3. TI-LFA backup paths traverse over adjacent (directly connected) P- and Q-nodes – and two additional MPLS backup labels (Node-SID associated with P-node, and Adj-SID associated with P-Q nodes adjacency) are required for protection

NOTE For link protection, TI-LFA with options 1 through 3 discussed so far provides full coverage in any arbitrary redundant network topology with symmetrical link metrics, requiring a maximum two additional MPLS labels for protection. For different types of protection (for example, node protection, SRLG protection, or fate sharing protection) with symmetric or asymmetric link metrics, as well as for link protection with asymmetric link metric, label stack with more than two labels might be required.

More Additional Labels for a TI-LFA Backup Path

The next example uses a topology shown in Figure 5.7.

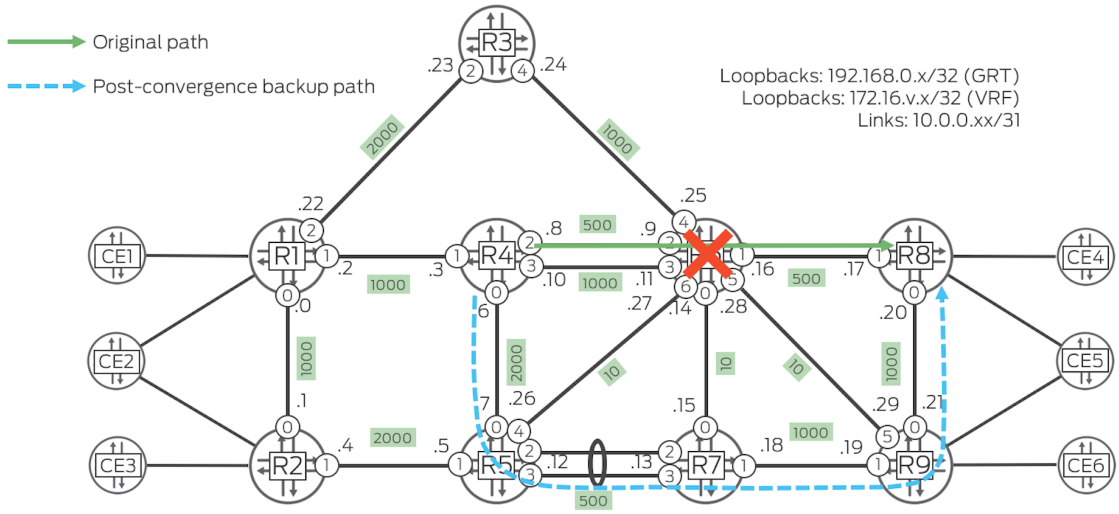

Figure 5.7 *TI-LFA with Node Protection*

TI-LFA with node protection is configured in the network. Investigating the routing state for the R8 loopback on router R4 you can notice an interesting observation:

```
kszarkowicz@R4# run show route 192.168.0.8 table inet.3 detail | match "entry|L-
ISIS|weight|operation"
192.168.0.8/32 (1 entry, 1 announced)
        *L-ISIS Preference: 14
                Next hop: 10.0.0.9 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Push 1408
                Next hop: 10.0.0.7 via ge-0/0/0.0 weight 0xf000
                Label operation: Push 90008, Push 70009, Push 50007(top)
```

The primary next hop has a single label (R8 Node-SID). However, the backup next hop this time has a label stack of 3 labels. These labels are actually unprotected Adj-SIDs (which were manually configured to have predictive values) for ISIS adjacencies on the TI-LFA backup path (for example, label 50007 is Adj-SID for the R5>R7 adjacency). It basically means, this time the backup path is encoded as the chain of three strict next hops, represented by three Adj-SIDs. If you look at the topology, paying special attention to the link metric, there is actually no other

possibility to encode a backup path using some loose next hops (Node-SIDs). Due to the low link metrics between R5/R7/R9 and R6, using loose next-hop (Node-SIDs) would result in the traffic being redirected back towards R6 node for any R5/R7/R9 node. This, however, due to configured node protection, should be avoided.

NOTE The TI-LFA draft doesn't provide any hints on how to determine required loop free label stacks for more advanced backup scenarios, like the topology shown in Figure 5.7. It is up to the software vendor to develop algorithms allowing for calculation of the label stack with minimal depth, while still fulfilling the loop-free requirement.

Protection for SR-TE Tunnels

SPRING machinery can provide protection not only for shortest path-based tunnels—what was discussed so far in this chapter—but as well for SPRING Traffic Engineered (SR-TE) tunnels.

Let's take as an example SR-TE tunnel shown in Figure 5.8 for reference.

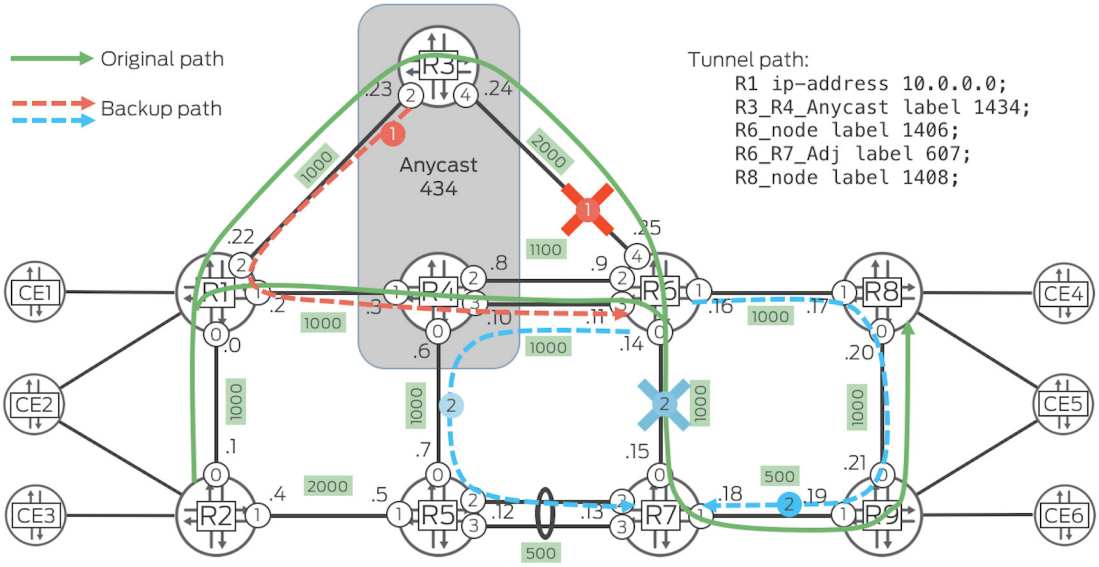

Figure 5.8 TI-LFA for SR-TE tunnel

This SR-TE tunnel is defined on R2 router with the following path:

- From R2, send the packet over direct link to R1 (IP-based adjacency).
- Next, from R1, send the packet over shortest path to any node advertising Anycast-SID 434 (Label 1434: Anycast-SID for R3/R4 nodes).
- Next, from Anycast-node, send the packet over shortest path to R6 (Label 1406: Node SID of R6 node).
- Next, from R6, send the packet over direct link to R7 (Label 607: Adj-SID for R6>R7 adjacency, with 'B' flag set).
- Next, from R7, send the packet over shortest path to R8 (Label 1407: Node SID of R8 node).

NOTE As opposed to RSVP based tunnels, there is no notion of *protection eligible* tunnel for SR-TE tunnels. Each SID specified during SR-TE tunnel definition, might be subject to protection or not. As such, any Prefix (Node, Anycast) SID is always subject to protection, while Adj-SID, depending if advertised with 'B' flag set or unset, might, or might not be subject to protection, as discussed in Chapter 2.

Let's investigate a couple of failure scenarios. All routers are configured to provide loose node protection. So first, each transit router tries to find a backup next hop protecting against upstream node failure, and if that's not possible, it tries to use a backup next hop protecting against upstream link failure.

On router R3, routing states associated with label 1406 (Node SID or R6 router) are as follows:

```
kszarkowicz@R3# run show route label 1406 detail | match "entry|weight|operation"
1406 (1 entry, 1 announced)
                Next hop: 10.0.0.25 via ge-0/0/4.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.22 via ge-0/0/2.0 weight 0xf000
                Label operation: Swap 1406
1406(S=0) (1 entry, 1 announced)
                Next hop: 10.0.0.25 via ge-0/0/4.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.22 via ge-0/0/2.0 weight 0xf000
                Label operation: Swap 1406
```

First of all, there are two entries for label 1406 (Node-SID of R6). The second entry has the marking S=0. It indicates that this entry handles packets with a *bottom-of-stack* (S) bit in MPLS header unset, implying that there are more MPLS labels in the packet. The first entry, on the other hand, is used to handling packets with a single MPLS label (1406) only. In this particular case, the content of both entries is equal.

Based on the SR-TE definition, packets belonging to the SR-TE tunnel in question will arrive at R3 with multiple labels (from top: 1406, 607, 1408, and some service labels), therefore the second FIB entry will be used to forward traffic. When analyzing the structure of the entry you will recognize that the protection is realized via R1 as a backup next hop, and the existing label 1406 is swapped to 1406; basically, the label is retained.

What is the end result? As a protection measure, the packet will be sent via R1 over the shortest path towards R6. Is this the desired outcome? It depends. For packets destined to R6 – yes, it is good. The packet is fast-rerouted to R6.

However, in the example in question, where an SR-TE tunnel is used, R6 is just a transit node for packets traversing the SR-TE tunnel. And, to protect against R6 transit node failure (remember, all routers in the topology are configured for loose node protection), it would be better to fast-reroute the traffic towards the next-next hop node (after R6) on the SR-TE path. In this particular case – towards node R7. How it could that be achieved? Simply, R3 should send the traffic to R1 (as we observe now), but instead of retaining label 1406, it should swap it to label 1407 (Node-SID of R7). From R1 the shortest path to R7 is via the R1>R4>R5>R7 path, therefore R6 is not touched. Node protection would be achieved.

But, as you can observe in the show output, R3 doesn't do this. Why? R3 is not able to determine next next-hop node (node after R6) based on just the top label (label 1406). This label simply informs you that the packet must be delivered to R6 over the shortest path. It doesn't contain any information, such as what should happen to the packet when it reaches R6. So R3 doesn't know if the next-next hop is, for example, R4 or R7, thus it is not able to determine the label (or label stack) that should be used to forward the packet towards the next-next hop, avoiding next hop.

So instead of using the post-converge path to the final destination of the SR-TE tunnel (or, rather, to the next-next hop on SR-TE tunnel, represented by the second label of the packet), the current means of implementing protection for SR-TE is to use a post-convergence path towards the next hop (represented by the top label on the packet) of the SR-TE tunnel.

Is it possible to make a better decision about determining backup for the next hop and associated label stack for SR-TE protection? Yes. There is an early draft (*drafthegde-spring-node-protection-for-sr-te-paths*) where the problem is discussed in more detail, though at the time of this writing it is not yet implemented in the Junos OS.

The idea of this draft is to perform a double MPLS lookup. The first lookup is to determine next hop. The second lookup, using the neighbor's context table (the RIB built from the perspective of the next hop node), is to determine the next-next hop. Once the next-next hop is determined, an appropriate backup path towards the next-next hop, avoiding the next hop, could be found and programmed into the FIB.

Now, let's have a look at the next failure case, which is R6>R7 link (or R7 node) failure. The packet arrives to R6 with R6>R7 Adj-SID (607) as the top label, therefore status for that label must be checked.

```
kszarkowicz@R6# run show route label 607 detail | match "entry|weight|operation"
607 (1 entry, 1 announced)
                Next hop: 10.0.0.15 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.17 via ge-0/0/1.0 weight 0xf000
                Label operation: Swap 1407
                Next hop: 10.0.0.10 via ge-0/0/3.0 weight 0xf000
                Label operation: Swap 1407
607(S=0) (1 entry, 1 announced)
                Next hop: 10.0.0.15 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.17 via ge-0/0/1.0 weight 0xf000
                Label operation: Swap 1407
                Next hop: 10.0.0.10 via ge-0/0/3.0 weight 0xf000
                Label operation: Swap 1407
```

There are no surprises. Again, only the backup paths towards next hop (R7) are installed but not towards the next-next hop (R8) . Due to the fact that in this given topology there are two ECMP backup paths, and the maximum ECMP backup was configured earlier in this chapter to 2, two backup next hops are installed.

Just for the record, there are two Adj-SIDs configured for R6>R7 adjacency – `protected` and `unprotected` – as follows:

```
protocols isis {
    interface ge-0/0/0
        level 2 ipv4-adjacency-segment {
            protected label 607;
            unprotected label 60007;
        }
    }
}
```

```
kszarkowicz@R2# run show isis database R6 extensive | find "IS extended neighbor: R7" | match Adj-SID
      P2P IPV4 Adj-SID - Flags:0x34(F:0,B:0,V:1,L:1,S:0,P:1), Weight:0, Label: 60007
      P2P IPv4 Adj-SID:   60007, Weight:   0, Flags: --VL-P
      P2P IPV4 Adj-SID - Flags:0x74(F:0,B:1,V:1,L:1,S:0,P:1), Weight:0, Label: 607
      P2P IPv4 Adj-SID:     607, Weight:   0, Flags: -BVL-P
```

In the previous example, you saw the SR-TE tunnel using the `protected (607) Adj-SID` for R6>R7 adjacency. But what would happen if the SR-TE tunnel uses the `unprotected (60007)` Adj-SID. Let's check:

```
kszarkowicz@R6# run show route label 60007 detail | match "entry|weight|operation"
60007 (1 entry, 1 announced)
                Next hop: 10.0.0.15 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Pop
60007(S=0) (1 entry, 1 announced)
                Next hop: 10.0.0.15 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Pop
```

Bingo! No backup next hops are installed. In such a case, this part of your SR-TE tunnel is not protected by TI-LFA.

Since this is the last example in the SR-TE protection section, let's have a look at the protection of Anycast-SID:

```
kszarkowicz@R1# run show route label 1434 detail | match "entry|weight|operation"
1434 (1 entry, 1 announced)
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0x1
                Label operation: Pop
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Pop
1434(S=0) (1 entry, 1 announced)
                Next hop: 10.0.0.3 via ge-0/0/1.0 weight 0x1
                Label operation: Pop
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Pop
```

Again, as already discussed in the previous chapter, this is no surprise here. Despite the fact there is no ECMP from R1 to R6 (the path to R6 via R3 has a cost of 3000, while the path via R4 has a cost of 2000) traffic is load-balanced between R3 and R4. This is due to the fact that R1 has ECMP towards Anycast-SID (cost of 1000 towards R3, and cost of 1000 towards R4).

Now let's verify the same SR-TE tunnel using a topology with slightly-changed link metrics, as illustrated in Figure 5.9.

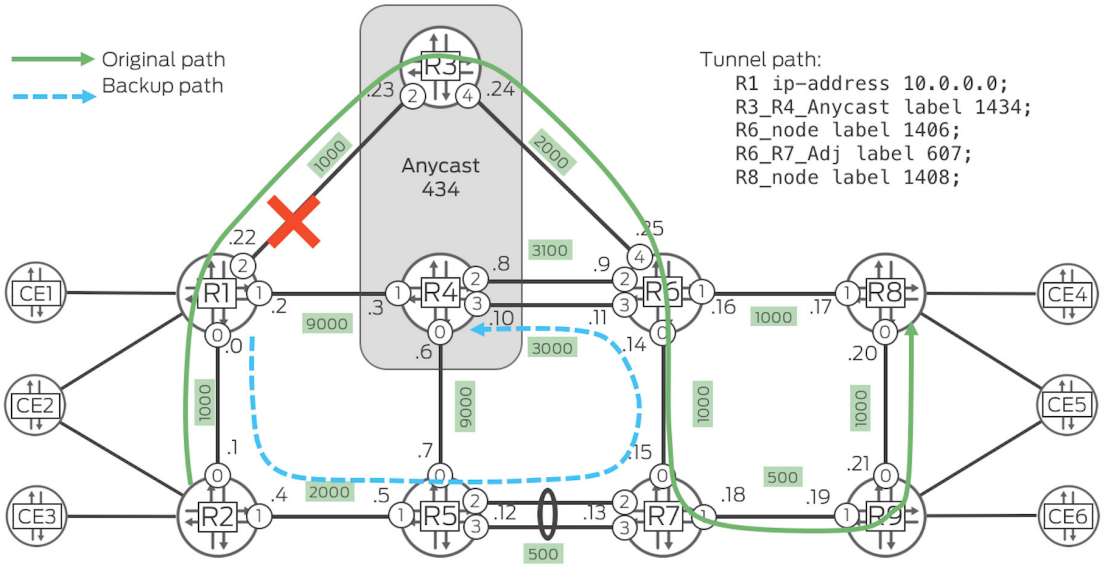

Figure 5.9 *TI-LFA for Anycast-SID*

This time, instead of ECMP, you see primary and backup next hops.

```
kszarkowicz@R1# run show route label 1434 detail | match "entry|weight|operation"
1434 (1 entry, 1 announced)
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000
                Label operation: Swap 1404
1434(S=0) (1 entry, 1 announced)
                Next hop: 10.0.0.23 via ge-0/0/2.0 weight 0x1, selected
                Label operation: Pop
                Next hop: 10.0.0.1 via ge-0/0/0.0 weight 0xf000
                Label operation: Swap 1404
```

Interestingly, the backup path is towards R4 (see label 1404, representing Node-SID of router R4, in label operation for backup next hop), and not towards R3.

In the end, this results in node protection: you can observe protection against R3 node failure, not only against R1>R3 link failure (remember, loose node protection is configured on the routers). In fact, with Anycast-SID, there is no need to investigate the second label to provide node protection. The backup path can simply point to a second node in the anycast group, and the lookup for the top label for such a decision is enough.

Is this backup path the shortest post-convergence path?

Not really. The shortest post-convergence backup path should point to R6 or R7 (depending on how we define the shortest post-convergence backup path for the SR-TE tunnel), with label operation 'swap 1406' (backup path towards R6) or label operation 'swap 1407' (backup path towards R6). However, to build such a backup path the router you would need to look up not only the top label of the received packet (here: Anycast-SID for R3/R4 nodes), but also the second top label (Node-SID of R6) to merge the backup path and the original SR-TE tunnel at R6, or even the third top label (the Adj-SID for R6>R7 adjacency) to merge at R7.

As discussed earlier in this chapter, multi-label lookup to build more optimized backup paths for SR-TE tunnels was not yet implemented in the Junos OS at the time this book was written.

Summary

This chapter has provided an example of how SPRING fast reroute mechanisms, called TI-FLA, are realized. TI-LFA provides a great framework for protection and can work in any arbitrary network topology.

More on Segment Routing

If you would like to read more about Segment Routing, in more detail, a good starting point is the Source Packet Routing in Networking (SPRING) working group in the IETF. You can see a list of the working group drafts and RFCs at https://datatracker.ietf.org/wg/spring/documents/.

The Juniper-sponsored J-Net Forum is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions within this large and vocal community. Be sure to register to participate at this free forum: https://forums.juniper.net.

The Juniper Networks TechLibrary has great segment routing documentation at: https://www.juniper.net/documentation/en_US/junos/topics/concept/source-packet-routing.html.

MPLS in the SDN Era, by Antonio Sánchez-Monge and Krzysztof Grzegorz Szarkowicz, (2015) Paperback, 900 pages, Publisher: O'Reilly Media, ISBN: 978-1-491-90545-6. http://shop.oreilly.com/product/0636920033905.do