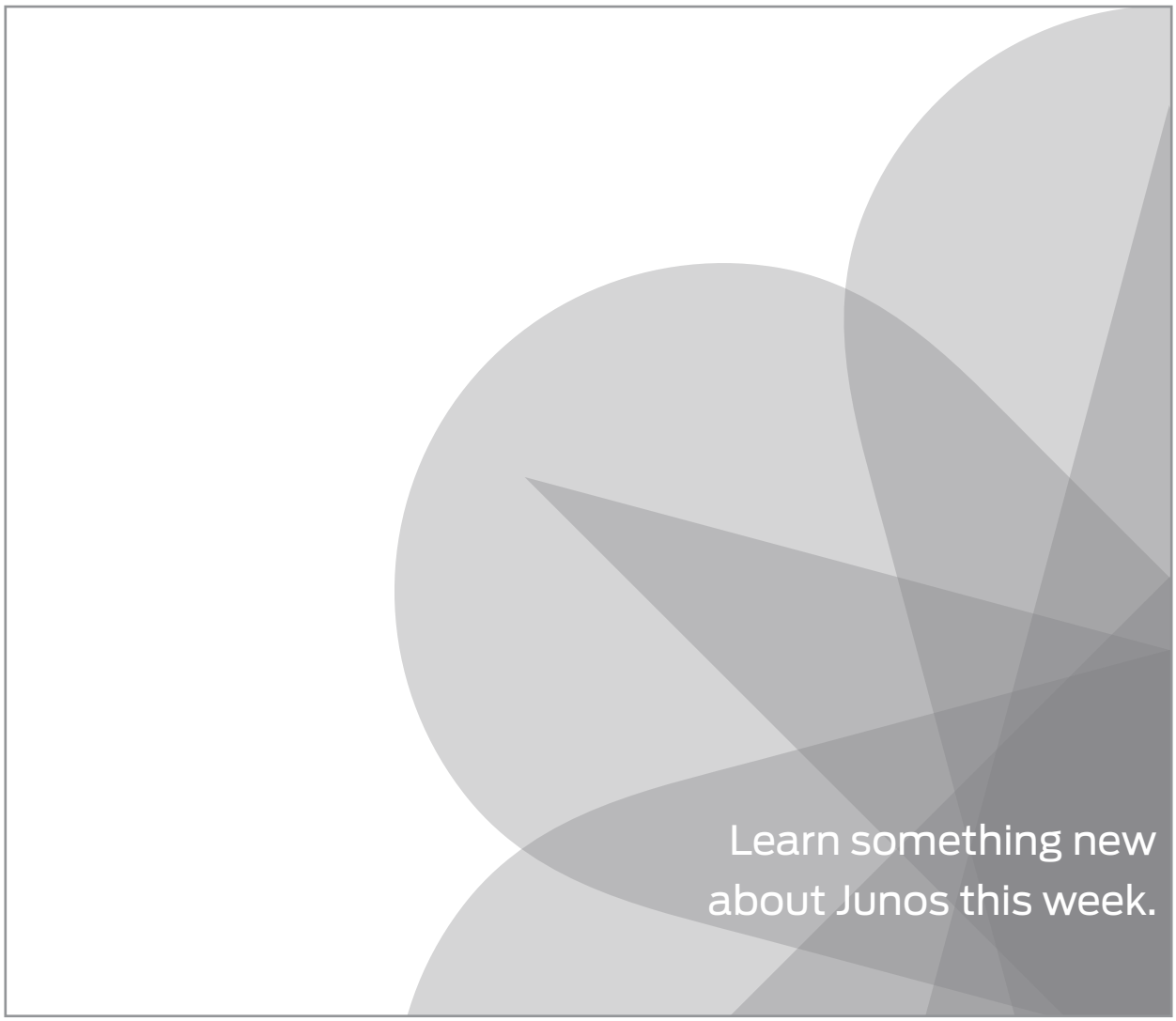
JUNIPER NETWORKS

Junos® Networking Technologies Series

# THIS WEEK: DEPLOYING MPLS




By Tim Fiola and Jamie Panagos

# THIS WEEK: DEPLOYING MPLS

While there are many books and papers available that cover network architecture, MPLS services, and MPLS cores, none put all these subjects together in a "beginning-to-end" walk-through methodology using all the necessary configuration examples for Juniper routers, with explanations for each configuration. *This Week: Deploying MPLS* is a seminar-in-a-book on the process of designing and standing up a MPLS core, as well as provisioning MPLS services such as L3VPN, VPLS, and Layer 2 circuits.

*This Week: Deploying MPLS* assumes readers have a working knowledge of OSPF or ISIS, iBGP, and eBGP, and have already made a choice as to which IGP to use in their network: OSPF or ISIS. These prerequisites clear the path for an elaborate walk-through deployment of a fault-tolerant MPLS network that includes the "how-to" Junos configurations along with the "why-to" explanations on why some architectural decisions are advantageous and others should be avoided.

If you are a network engineer, network architect, or network administrator in an enterprise or service provider environment that has decided to implement MPLS, be prepared to be shown, not told, what to do.

"If you're thinking about adding MPLS to your Junos network, this book is perfect. Not only will you find MPLS concepts explained but many real-world Junos configuration examples, too. In short, the book provides the necessary knowledge for an MPLS deployment in a matter of days because you're never more than a step away from configuring the concepts you've just learned."

Nicholas Harland, Senior Network Engineer, Constant Contact, Inc.

## LEARN SOMETHING NEW ABOUT JUNOS THIS WEEK:

- Decide on an appropriate network architecture based on your network's requirements and offered services.
- Implement a fault-tolerant MPLS core based on LDP, RSVP, or LSP and RSVP, on Juniper Networks routers using the Junos CLI.
- Understand how to troubleshoot an MPLS core.
- Provision L3VPN, VPLS, and Layer 2 circuits on Juniper Networks routers.
- Understand all the Junos features in an MPLS implementation.
- Effectively implement traffic engineering and understand how to effectively and efficiently scale your network.

Published by Juniper Networks Books
www.juniper.net/books

ISBN 9367792468

9 789367 792469 52400

07500210

# Junos® Networking Technologies

## *This Week: Deploying MPLS*

By Tim Fiola and Jamie Panagos



JUNIPER® NETWORKS

**Published by Juniper Networks Books**
Writers: Tim Fiola, Jamie Panagos
Technical Review: Eural Authement
Editor in Chief: Patrick Ames
Proofreader: Nancy Koerbel
Junos Program Manager: Cathy Gadecki
J-Net Community Management: Julie Wider

**About the Authors**
Tim Fiola is a Senior Network Consultant in Juniper Networks' Professional Services Organization, specializing in the design, implementation, and operations of MPLS networks. He is JNCIE-M/T #419, is JNCIS-SEC certified, and has over 7 years experience working with Junos devices.

Jamie Panagos is a Senior Network Consultant in Juniper Networks' Professional Services Organization and specializes in the design, implementation,º and operation of datacenter, enterprise, and service provider networks. Jamie has over 10 years of experience on some of the largest networks in the world and has participated in several influential industry communities including NANOG, ARIN and RIPE. He holds JNCIE-M/T #445 and JNCIE-ER #50.

**Authors' Acknowledgments**
Tim Fiola and Jamie Panagos would like to thank Patrick Ames and Cathy Gadecki for their hard work in bringing the idea of this book to fruition. They would also like to thank Craig Sirkin for the schedule flexibility that helped make the writing of this book possible. Finally, special thanks to Eural Authement, who very kindly gave up time in his very busy schedule to technically edit this book, and to our reviewers, Nick Harland of Constant Contact ® as well as Nick Slabakov and Nathan Alger of Juniper Networks, for their valuable feedback and insight.

Tim Fiola would like most of all, to thank his family; their support has made everything he has possible.

Jamie Panagos would also like to thank his family for their ongoing support and patience, which allow projects like this to be successful.

This book is available in a variety of formats. For more information see www.juniper.net/dayone.

Send your suggestions, comments, and critiques by email to: dayone@juniper.net.

Be sure to follow this and other Juniper Networks Books on: Twitter: @Day1Junos

Version History: v1 (This Week) April 2011

ISBN: 978-1-936779-24-6 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-936779-25-3 (ebook)

2 3 4 5 6 7 8 9 10 #7500210-en

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

## Welcome to *This Week*

*This Week* books are an outgrowth of the extremely popular *Day One* book series published by Juniper Networks Books. *Day One* books focus on providing just the right amount of information that you can do, or absorb, in a day. On the other hand, *This Week* books explore networking technologies and practices that in a classroom setting might take several days to absorb. Both book series are available from Juniper Networks at: www.juniper.net/dayone.

*This Week* is a simple premise – you want to make the most of your Juniper equipment, utilizing its features and connectivity – but you don't have time to search and collate all the expert-level documents on a specific topic. *This Week* books collate that information for you, and in about a week's time, you'll learn something significantly new about Junos that you can put to immediate use.

*This Week* books are written by Juniper Networks subject matter experts and are professionally edited and published by Juniper Networks Books. They are available in multiple formats, from eBooks to bound paper copies, so you can choose how you want to read and explore Junos, be it on the train or in front of terminal access to your networking devices.

## What You Need to Know Before Reading

Before reading this book, you should 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 the Junos configuration.

Other things that you will find helpful as you explore the pages of this book:

- Having access to a Junos device while reading this book will allow you to test configurations and your own ideas. A Copy and Paste edition of this book is freely available at www.juniper.net/dayone, in rich text format, for easy copying and pasting of the book's configurations into your favorite text editor.
- A basic understanding of routing and routing protocols including BGP, OSPF, and ISIS.

## Assumptions

This book makes a few assumptions about you. If you do not meet any one of the following criteria, the book may harder to digest and the configuration samples more difficult to implement on your device or test bed.

- You have practical experience and at least a basic working knowledge of BGP and either OSPF or ISIS.
- You have a good working knowledge of the Junos CLI.
- You are interested in or have a plan to deploy MPLS.
- Your initial deployment will focus on unicast, IPv4 traffic within your routing domain.

## After Reading This Book You'll Be Able To

This book helps you to understand the mechanics and features of multi-protocol label switching (MPLS). This includes how MPLS provides value, how to design and implement an MPLS network, and how to enable MPLS applications such as Layer 3 VPN and VPLS. You will learn specific MPLS skills in this book, and when you're done you'll be able to:

- Understand what a label-switched path is and how it works.
- Explain the differences between and LDP and RSVP.
- Understand resiliency in an MPLS network.
- Implement an MPLS network, including advanced features such as traffic-engineering and failure recovery.
- Understand how to configure basic implementations of Layer 3 VPNs, Layer 2 VPNs, and VPLS.
- Understand the factors to consider in network design and architecture.
- Analyze scaling factors when designing and growing an MPLS network.
- Use example networks to configure your own network.

## Special Notes to the Reader About Deploying MPLS

There were two test beds used in this book. The first was a test bed consisting of J-series routers in packet mode. The J-series routers running VPLS ran Junos 10.0R3.10; the routers not running VPLS used 9.6R4.4.

To demonstrate some MPLS features only available (as of this writing) on high-end routers and the MX series platform, a testbed consisting of MX-80, M10i, and M5 routers running Junos 10.3R1.9 was used.

This book does not discuss some advanced MPLS topics such as inter-AS VPNs, NG-MVPN, and IPv6 as the goal was to provide an introduction to MPLS concepts and examples of common deployments. The hope is that you can use what you learn in this book to successfully deploy an MPLS network and build on that knowledge when you require advanced topics such as those mentioned above.

MORE? An excellent source for NG-MVPN is the new *This Week: Deploying MBGP Multicast VPNs*, available at www.juniper.net/dayone.

MORE? An excellent source of advanced MPLS topics can be found in *MPLS-Enabled Applications, Third Edition*, by Ina Minei and Julian Lucek (2011, Wiley & Sons Publishers). You can get more information on this best-selling MPLS book at www.juniper.net/books.

# Chapter 1

## MPLS Core Network Concepts

Once the decision to implement a Multi-Protocol Label Switching (MPLS) core with MPLS services has been made, it can be a daunting process to understand the concepts, design a flexible and scalable architecture that meets your specific needs, and then, finally, implement the design into the network. This book guides the reader through those steps and this first chapter explains the different MPLS core network concepts. With a solid understanding of the concepts, you'll be able to confidently move forward and make informed decisions about what traits your new Junos MPLS core network will have and what it may need.

## Introduction

One of the most important choices for an MPLS network is the core MPLS protocol. Label-switched-paths (LSPs) require an MPLS protocol on each router that a LSP traverses. An MPLS protocol allows a router to establish a local database that matches labels with destinations and other labels, exchange those labels with neighboring routers, and send and receive labeled packets. This chapter covers the choices for MPLS protocols, and later, Chapter 3 covers how to determine which MPLS protocol is best for a specific network, whether it is RSVP, LDP, or both. For now, let's start by defining what an LSP is and how an LSP is created.

A LSP provides a route through the network for a labeled packet. Instead of routing based on the packet's destination IP address, a labeled packet is routed through the network based on the value of a numerical label attached to the packet. A label edge router (LER) is an LSP's ingress or egress router. A pure label-switching router (LSR), on the other hand, is a *transit* router for LSPs that switches *solely* on the basis of the incoming label's instructions.

Figure 1.1 shows five MPLS routers and an LSP from R1 to R5: R1 is the LSP's ingress LER, R2, R3 and R4 are LSRs, since they only switch based on MPLS labels, and R5 is the LSP's egress LER. R4 advertises a label to R3, notifying R3 that R5 can be reached via R4 if R3 adds label 234875 to the packet it sends to R4 (the process of adding a label to a packet's label stack is called a *push*). This process continues, with each router advertising to its neighbor how to reach R5 via that neighbor. The series of labels along the path of routers that R1 uses to reach R5 is the *label-switched-path* from R1 to R5. Because a unique LSP has a unique combination of labels, transit routers, and destination, an LSP is unidirectional in nature. So the LSP that takes traffic from R1 to R5 in Figure 1.1 is *not* the same LSP as the one that takes traffic from R5 to R1.

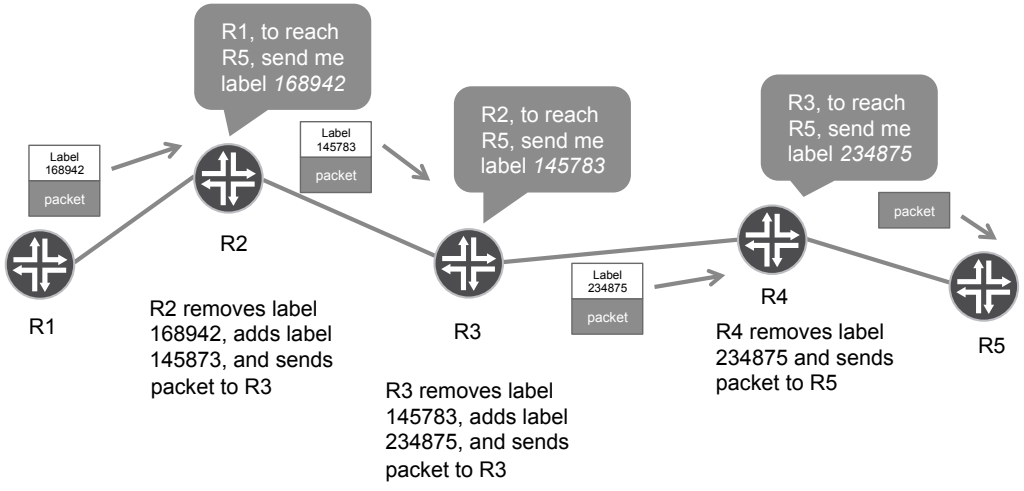



Figure 1.1 MPLS Label Propagation for R5

NOTE A packet's *label stack* refers to the labels that are prepended to the packet. When drawn graphically, it often appears as a packet with one or more labels stacked on top of it. Later chapters in this book examine why stacking multiple labels is often necessary.

In Figure 1.1, R1 pushes a label on to the packet and forwards the packet on to R2. R2 receives the labeled packet, removes the 168942 label, replaces it with the 145783 label (this process is called a label *swap*), and forwards it on to R3. R3 performs a similar label swap, switching label 145783 for 234875. Notice that R4 receives a labeled packet from R3, but then removes the label (removal of a label from the label stack without replacing it is called a *pop*) and forwards the packet to R5 without a label. This is called *penultimate hop popping* (PHP). Since R5 is at the tail-end of the LSP, it has no need to receive a labeled packet. PHP allows R5 to skip the step of popping the label before forwarding the packet based on destination address.

When an LSP's egress router tells the penultimate router to perform penultimate hop popping, it advertises a label object with a value of 3, telling the router to pop the top label and send the packet. The label value 3 is a value reserved exclusively for *notification* of penultimate hop popping (the label value of 3 is not actually transmitted in a label). Below, for the *lagavulin-to-dalwhinnie* LSP, glenlivet is the penultimate router and it shows a label out value 3 for the LSP traffic bound for dalwhinnie – meaning it sends the packet to dalwhinnie without the LSP label. Upon receipt of the packet with no label, the only operation dalwhinnie need do is perform a normal route lookup on the destination address.

```
ps@glenlivet> show mpls lsp name lagavulin-to-dalwhinnie detail
Ingress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Egress LSP: 2 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 1 sessions

10.200.86.5
  From: 10.200.86.7, LSPstate: Up, ActiveRoute: 1
  LSPname: lagavulin-to-dalwhinnie, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 3
  Resv style: 1 FF, Label in: 300016, Label out: 3
  Time left:  126, Since: Tue Feb 15 03:28:02 2011
  Tspec: rate 64kbps size 64kbps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 30254 protocol 0
  PATH rcvfrom: 192.168.86.45 (ge-0/0/3.0) 42150 pkts
  Adspec: received MTU 1500 sent MTU 1500
  PATH sentto: 192.168.86.6 (ge-0/0/2.0) 42297 pkts
  RESV rcvfrom: 192.168.86.6 (ge-0/0/2.0) 42264 pkts
  Explct route: 192.168.86.6
  Record route: 192.168.86.2 192.168.86.42 192.168.86.45 <self> 192.168.86.6
Total 1 displayed, Up 1, Down 0
```

PHP is the default behavior for any Junos device running MPLS. The alternative to PHP is *ultimate hop popping* (UHP). Ultimate hop popping causes the LSP's egress router (the ultimate router in the LSP) to request that the LSP's penultimate router send to it a labeled packet. If UHP is configured in Figure 1.1, R4, the penultimate router, performs a label *swap*, instead of a *pop*, sending a labeled packet to R5.

NOTE UHP may be required in some MPLS networks for class-of-service to work properly or for vendor interoperability.

To configure UHP on a Junos device configure *explicit-null* under the MPLS protocol stanza, like this:

```
[edit protocols mpls]
ps@dalwhinnie# show
explicit-null;
. . .
. . .
```

If an egress router is configured for ultimate hop popping, it advertises a label object with a value of *0* to the penultimate router. This notifies the penultimate router to perform a label swap (instead of a pop) on the top label, sending the packet to the egress router with a top label value of 0 (see Figure 1.2). The label value 0 is reserved for ultimate hop popping. Below, dalwhinnie is now configured for UHP; glenlivet, the penultimate router, now shows a label out value 0. Dalwhinnie then pops the label and does a normal routing table lookup.

```
ps@glenlivet> show mpls lsp name lagavulin-to-dalwhinnie detail
Ingress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Egress LSP: 2 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 1 sessions

10.200.86.5
  From: 10.200.86.7, LSPstate: Up, ActiveRoute: 1
  LSPname: lagavulin-to-dalwhinnie, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 0
  Resv style: 1 FF, Label in: 302016, Label out: 0
  Time left:  146, Since: Tue Feb 15 03:28:02 2011
  Tspec: rate 64kbps size 64kbps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 30254 protocol 0
  PATH rcvfrom: 192.168.86.45 (ge-0/0/3.0) 42156 pkts
  Adspec: received MTU 1500 sent MTU 1500
  PATH sentto: 192.168.86.6 (ge-0/0/2.0) 42302 pkts
  RESV rcvfrom: 192.168.86.6 (ge-0/0/2.0) 42269 pkts
  Explct route: 192.168.86.6
  Record route: 192.168.86.2 192.168.86.42 192.168.86.45 <self> 192.168.86.6
Total 1 displayed, Up 1, Down 0
```

NOTE The IPv6 explicit null label has a reserved value of 2.

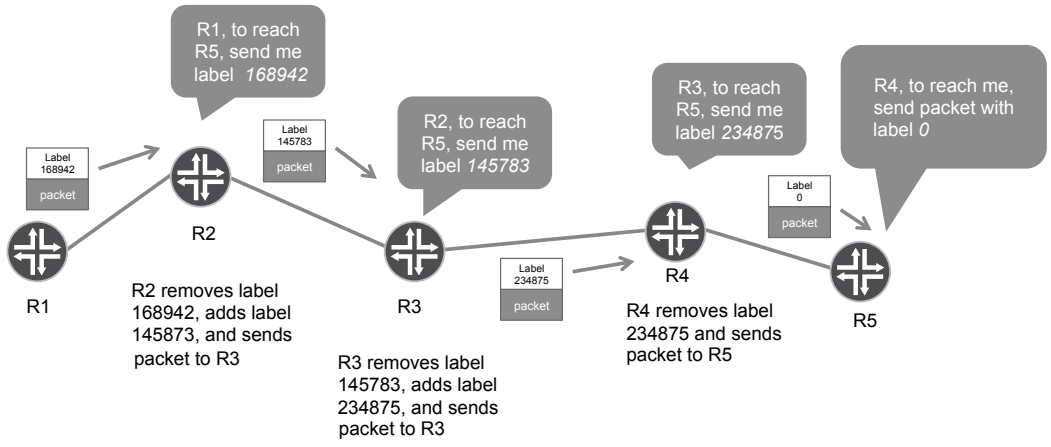



Figure 1.2 MPLS Label Propagation for R5 Using UHP

## RSVP LSPs

Before proceeding further, and for future reference, Figure 1.3 is the network layout for the nine routers in the MPLS domain referenced in the rest of this book. This diagram does not yet include any customer edge routers, as those will be introduced when Chapter 2 discusses MPLS services. Unless noted otherwise, each of these routers is running both RSVP and LDP as the core MPLS protocols on each core-facing interface – refer back to this drawing for the case studies in this chapter.

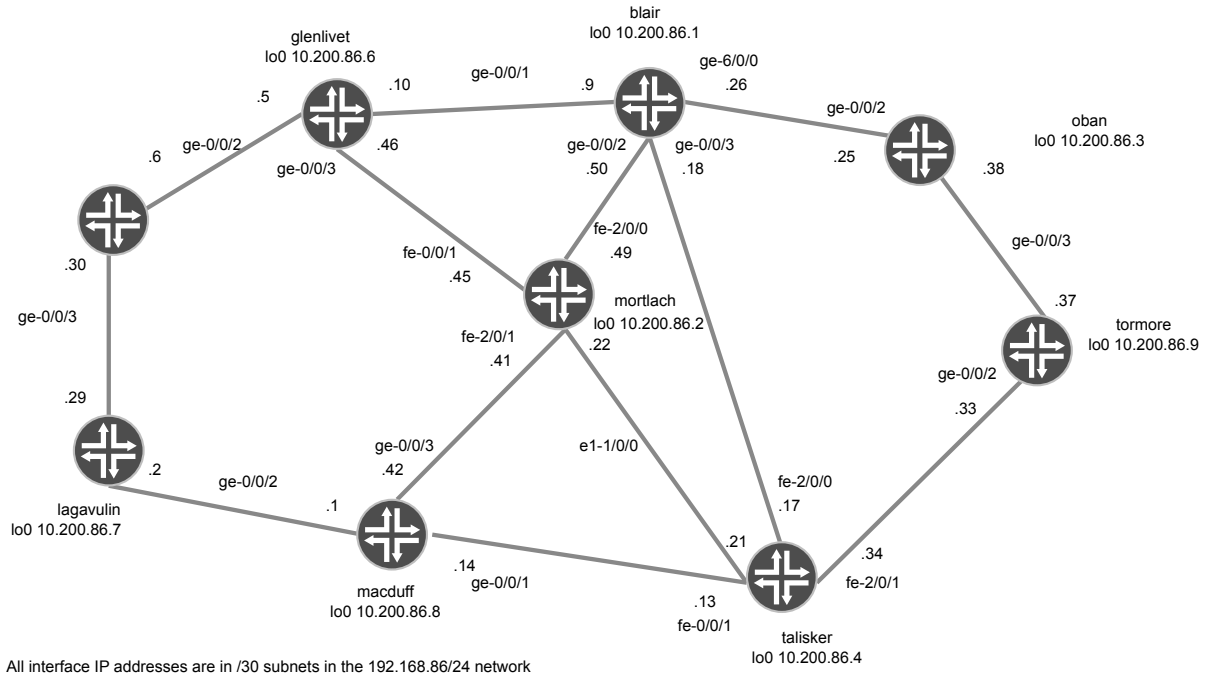



Figure 1.3 This Book's Network Layout of Nine Routers in the MPLS Domain

RSVP is short-hand for *resource reservation protocol*. RSVP is a common choice for a core network MPLS protocol because of its rich variety of features including traffic engineering (TE), fast-failover, quality-of-service, bandwidth reservation, and LSP customization. RSVP LSPs require more configuration than LDP LSPs but have the ability to provide more features than an LDP LSP (LDP is covered in the following section). Recent features such as RSVP auto-mesh (available starting in Junos 10.1) do reduce the configuration complexity for RSVP. RSVP auto-mesh is discussed in Chapter 3.

*To Configure a Basic RSVP LSP on a Junos Device*

Start with the LSP's ingress router.

1. Configure the interfaces that will be running RSVP with *family mpls* (these are typically all of the core-facing interfaces):

```
 [edit interfaces]
ps@dalwhinnie# show
 . . .
 <snip>
 . . .
ge-0/0/2 {
    unit 0 {
        description "Connection to glenlivet ge-0/0/2.0";
        family inet {
            address 192.168.86.6/30;
        }
        family mpls;
    }
}
ge-0/0/3 {
    unit 0 {
        description "Connection to lagavulin ge-0/0/3.0";
        family inet {
            address 192.168.86.30/30;
        }
        family mpls;
    }
}
```

2. Configure a *label-switched-path* with a `to <destination>` statement and configure the interfaces that will be running MPLS under the `[edit protocols mpls]` stanza:

```
 [edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban { ##only on lsp ingress router
    to 10.200.86.3;
}
interface ge-0/0/2.0;
interface ge-0/0/3.0;
```

3. Configure the interfaces that will be running RSVP under the `[edit protocols rsvp]` stanza:

```
 [edit protocols rsvp]
ps@dalwhinnie# show
interface ge-0/0/2.0
interface ge-0/0/3.0
```

Now configure each router in the MPLS domain with steps 1-3, *minus* the `label-switched-path` statement shown in step 2. An RSVP LSP is only configured on the LSP's ingress router.

There is an often-asked question regarding steps (1) and (2) above: Why does the configuration require interfaces to be specified under the `[edit protocols mpls]` level and the same interfaces to have the *family mpls* configuration at the `[edit interfaces unit <unit number>]` level? Isn't that redundant?

It's a great question and here's why it's done. Specifying interfaces under `[edit protocols mpls]` allows those interfaces to run the MPLS protocol and appear in the Traffic Engineering Database (TED) as possible resources for use by RSVP LSPs. Configuring *family mpls* on the logical interface, on the other hand, allows that interface to send and receive labeled packets – without this ability, the interface does not know what to do with a labeled packet and simply drops it. It is a common mistake to configure one and not the other and then spend time troubleshooting why

the configured RSVP LSP's all show *CSPF Failed: no route toward <destination>*, as shown here:

```
[edit protocols mpls]
ps@dalwhinnie# show label-switched-path dalwhinnie-to-lagavulin
to 10.200.86.7;

[edit protocols mpls]
ps@dalwhinnie# run show mpls lsp name dalwhinnie-to-lagavulin detail
Ingress LSP: 3 sessions

10.200.86.7
  From: 0.0.0.0, State: Dn, ActiveRoute: 0, LSPname: dalwhinnie-to-lagavulin
  ActivePath: (none)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary                State: Dn
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Will be enqueued for recomputation in 19 second(s).
   84 Oct  2 00:51:48.731 CSPF failed: no route toward 10.200.86.7[9 times]
```

Notice that the RSVP LSP *dalwhinnie-to-lagavulin* is failing to come up because there is no complete MPLS-enabled path to the LSP's destination.

NOTE LDP does not require configuring interfaces under `[edit protocols mpls]`. Only RSVP LSPs require that configuration. Both protocols, however, require that *family mpls* be configured on the actual logical interfaces.

NOTE It's a bit counterintuitive, but RSVP-signaled LSPs are configured under the `[edit protocols mpls]` hierarchy.

Once everything is configured correctly, the RSVP LSP comes up. The following output shows the LSP state as *Up* and shows the LSP's *record route* (the LSP's path through the network). The LSP's received record route contains the list of interface IP addresses that the RSVP LSP travels through, including the ingress and egress nodes. The output also shows that this router is pushing label 300432 on to the packet before sending it to the downstream router:

```
ps@dalwhinnie> show rsvp session lsp name dalwhinnie-to-oban detail
Ingress RSVP: 1 sessions

10.200.86.3
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 0
  LSPname: dalwhinnie-to-oban, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300432
  Resv style: 1 FF, Label in: -, Label out: 300432
  Time left:   -, Since: Wed Jul 21 03:36:04 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 1773 protocol 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.5 (ge-0/0/2.0) 126 pkts
  RESV rcvfrom: 192.168.86.5 (ge-0/0/2.0) 124 pkts
  Explct route: 192.168.86.5 192.168.86.9 192.168.86.25
  Record route: <self> 192.168.86.5 192.168.86.9 192.168.86.25
Total 1 displayed, Up 1, Down 0
```

Each router in the LSP's path (including ingress and egress routers) must have RSVP enabled on any interfaces that should be available for an RSVP LSP to use. This book explores additional RSVP capability in more detail in later chapters.

## LDP LSPs

Label distribution protocol (LDP) is another option for use as the core MPLS protocol. LDP is much easier to configure than RSVP, but it also lacks capabilities such as a fast-failover mechanism, traffic-engineering capability, and many other features. In short, LDP offers a relatively easy way to establish an MPLS core that can support MPLS services, but lacks abilities to do much more than ISIS or OSPF in the way of other features. On the other hand, recent developments, like LFA (Loop-free alternates) add some measure of failover mechanisms, which are still more limited than what RSVP provides.

### *To Establish an LDP Core*

1. Configure *family mpls* on core-facing interfaces on each router:

```
ps@tormore> show configuration interfaces
 . . .
 <snip>
 . . .
ge-0/0/2 {
    unit 0 {
        description "Connection to talisker fe-2/0/1.0";
        family inet {
            address 192.168.86.33/30;
        }
        family mpls;
    }
}
ge-0/0/3 {
    unit 0 {
        description "Connection to oban ge-0/0/3.0";
        family inet {
            address 192.168.86.37/30;
        }
        family mpls;
    }
}
```

2. Configure each interface in Step 1 under the `[edit protocols ldp]` stanza

```
ps@tormore> show configuration protocols ldp
interface ge-0/0/2.0;
interface ge-0/0/3.0;
```

3. Repeat Steps 1-2 on each router in the MPLS domain

When Steps 1-3 are complete, each router can exchange LDP labels with its neighbors for the route to its lo0 address. Every neighbor router that receives that label then advertises that LDP reachability to its other neighbors via a label, and so on, resulting in each router having an LDP LSP to the lo0 address for every other router in the LDP domain. Unlike RSVP, an LDP LSPs route through the network cannot be directly controlled (it follows the IGP shortest path) or viewed from a single router. The LDP must be checked at each next-hop router to determine the label operation and the next router in the LSP:

```
 [edit protocols mpls]
ps@tormore# run show route protocol ldp 10.200.86.7

inet.0: 32 destinations, 32 routes (32 active, 0 holddown, 0 hidden)

inet.3: 19 destinations, 34 routes (8 active, 0 holddown, 19 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.7/32     *[LDP/9] 00:00:17, metric 1
                  > to 192.168.86.34 via ge-0/0/2.0, Push 300080
```

Tormore pushes a label with value 300080 onto a packet bound for 10.200.86.7.

## Failover

How the network behaves if a link or a node fails is one of the most important considerations for any network architect and network engineer. Among other things, proper design of failover mode helps ensure that capacity planning can adequately plan and augment the network as required and that quality of service requirements can be met. It is also a major factor in determining how network operations staff respond to a network failure. Juniper Networks routers provide a variety of options to cope with a failure in the network. The failover options discussed here are designed to protect RSVP LSPs against any single point of failure between the ingress and egress routers. Note that if there is more than one failure, the methods below may not work. Additionally, none of the methods discussed here protect against a catastrophic failure of an LSP's ingress or egress routers.

### Link Protection

Link protection provides protection against a link failure along an RSVP label-switched path. When link protection is configured, each router along the LSP (except for the egress router) attempts to find an alternate path to the next router in the LSP. This alternate path is known as a *next-hop bypass LSP*. The next-hop bypass LSP's purpose is to provide an alternate path to the router on the other side of the protected link. Each next-hop bypass LSP is established after the main LSP is set up. When a link failure along the LSP occurs between two routers, the repair action is initiated by the local router with the failed link that is closest to the LSP ingress router. This router is known as the *point of local repair* (PLR).

Figure 1.4 shows a link break between glenlivet and blair. Glenlivet is the router closer to the ingress router (dalwhinnie) and so glenlivet is the PLR. Since glenlivet and blair will be the first routers to know about the link break, glenlivet acts quickly to send traffic through the pre-signaled next-hop bypass LSP that runs from glenlivet-mortlach-blair. If network topology is not rich enough to provide for a specific next-hop bypass LSP, then a router does not create a bypass LSP. However, the router still supports the primary LSP.

The ability to quickly detect and act on a link break by the router closest to the break makes link protection the option with the quickest failover. Another benefit of link protection is its scalability. Glenlivet can use the single next-hop bypass LSP glenlivet-mortlach-blair to protect traffic on any LSP that transits its link to blair. This means that only a relatively small number of next-hop bypass LSPs need to exist in order to protect a large number of primary LSPs.

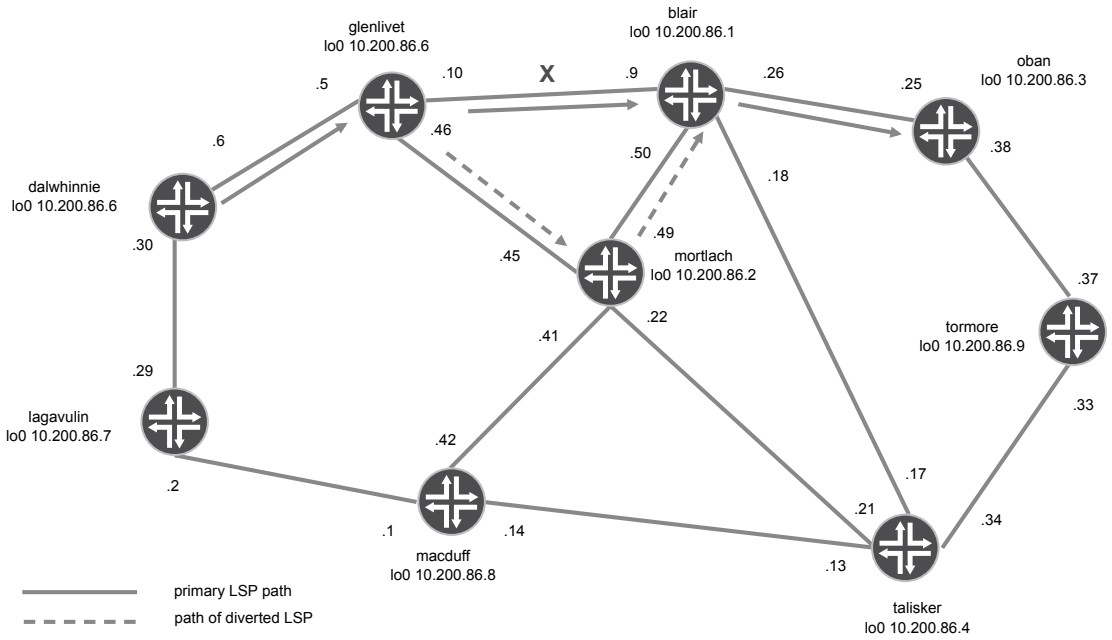



Figure 1.4 Link Protection on a Link Break

In Figure 1.5, the *dalwhinnie-to-oban* and *dalwhinnie-to-tormore* LSPs each share the link between glenlivet and blair. If that link goes down, both LSPs can take the next-hop bypass LSP through glenlivet-mortlach-blair. This capability to multiplex protected LSPs makes link-protection an extremely scalable LSP protection method.

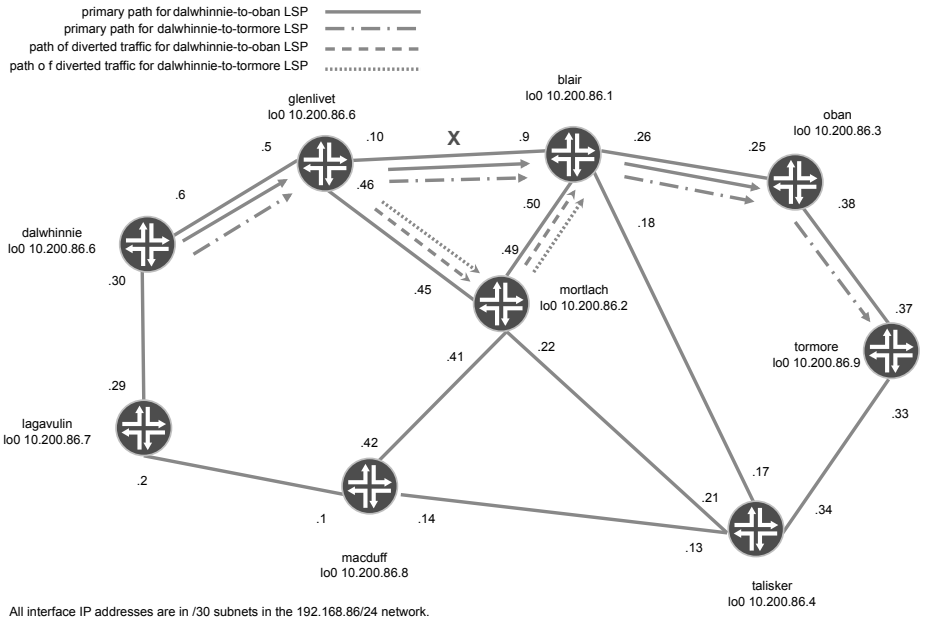



Figure 1.5 Dalwhinnie-to-oban and dalwhinnie-to-tormore LSPs Sharing Same Bypass LSP

The following output shows that a single bypass LSP on glenlivet protects two transit LSPs:

```
ps@glenlivet> show mpls lsp transit
Transit LSP: 2 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.3    10.200.86.5    Up      1  1 SE  300208   300112 dalwhinnie-to-oban
10.200.86.9    10.200.86.5    Up      1  1 SE  300224   300128 dalwhinnie-to-tormore
Total 2 displayed, Up 2, Down 0

ps@glenlivet> show mpls lsp bypass ingress detail
Ingress LSP: 1 sessions

10.200.86.1
  From: 10.200.86.6, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass->192.168.86.9
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 299936
  Resv style: 1 SE, Label in: -, Label out: 299936
  Time left:   -, Since: Fri Jul 16 04:08:48 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 53500 protocol 0
  Type: Bypass LSP
    Number of data route tunnel through: 2
    Number of RSVP session tunnel through: 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.45 (ge-0/0/3.0) 218 pkts
  RESV rcvfrom: 192.168.86.45 (ge-0/0/3.0) 218 pkts
  Explct route: 192.168.86.45 192.168.86.50
  Record route: <self> 192.168.86.45 192.168.86.50
Total 1 displayed, Up 1, Down 0
```

Glenlivet's bypass LSP *Bypass->192.168.86.9* is protecting two LSPs (*Number of data route tunnel through: 2*) and transits 192.168.86.45 and 192.168.86.50.

Once the traffic from the LSP reaches blair, it continues on the original LSP's signaled path to the respective oban or tormore destination. Traffic traveling on the next-hop bypass LSP is a short-term fix. When the PLR switches traffic to the bypass LSP, it also signals back to the LSP's ingress router. The ingress router then attempts to signal a different primary path for the LSP.

Understanding the label operations that occur when LSP traffic fails over to a bypass LSP provides great insight into the mechanics of the failover. Examining the *dalwhinnie-to-oban* LSP as it transits glenlivet, the router *swaps* label *303168* for *303296* before sending the traffic on to blair. In the event that the glenlivet-blair link fails, glenlivet switches the traffic to the bypass LSP, performs that same initial label swap and *then a push*, adding *300880* as the top label before forwarding the packet to mortlach. In this example, the *300880* label is sometimes referred to as a *bypass* label, since it's added when the traffic takes the bypass LSP. Let's examine the LSP:

```
ps@glenlivet> show mpls lsp transit name dalwhinnie-to-oban
Transit LSP: 5 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.3    10.200.86.5    Up      1  1 SE  303168   303296 dalwhinnie-to-oban
Total 1 displayed, Up 1, Down 0

ps@glenlivet> show route table mpls.0 label 303168 detail | match Label
             Label-switched-path dalwhinnie-to-oban
             Label operation: Swap 303296
```

```
            Label-switched-path Bypass->192.168.86.9
            Label operation: Swap 303296, Push 300880(top)
```

The output confirms that glenlivet receives a packet with label 303168 and swaps that label for 303296. In the event of a link break between glenlivet and blair, the output shows that glenlivet still swaps label 303168 for label 303296, but then it also pushes label 300880 on top of 303296 and sends the packet out the egress interface for the bypass LSP, knowing that the bypass LSP will deliver the packet to blair.

In the event of a link break, the packet arrives at mortlach, the router *pops* the *300880* label (the bypass LSP employs PHP), sending the packet to blair with only one label in the stack (*303296*):

```
ps@mortlach> show route table mpls.0 label 300880 detail

mpls.0: 27 destinations, 27 routes (27 active, 0 holddown, 0 hidden)
300880 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 584
              Next-hop reference count: 3
              Next hop: 192.168.86.50 via fe-2/0/0.0 weight 0x1, selected
              Label-switched-path Bypass->192.168.86.9
              Label operation: Pop
              State: <Active Int>
              Local AS:  7176
              Age: 54:59     Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I

300880(S=0) (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 594
              Next-hop reference count: 2
              Next hop: 192.168.86.50 via fe-2/0/0.0 weight 0x1, selected
              Label-switched-path Bypass->192.168.86.9
              Label operation: Pop
              State: <Active Int>
              Local AS:  7176
              Age: 54:59     Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I
```

NOTE The (S=0) entry in the mpls.0 table for a given label refers to a packet coming into the router with a label stack depth n>=2, and exiting the router with a label stack depth of n-1. The information without the *S=0* note refers to a packet entering the router with a label stack depth of 1 and exiting the router with no label. In this example, the packet enters the router with a stack depth n=2.

The net effect of the bypass LSP is that the packet arrives at blair with a top label of *303296*, which is the exact same situation as if the packet had not taken the bypass LSP (the only difference being that the packet arrives on a different interface, which is fine because the labels are not tied to a specific interface) as shown in Figure 1.6. This means that blair does not have to modify its processing behavior, even though it's receiving traffic on a bypass LSP. Both link-protection and node-link-protection (discussed in the next section) use bypass labels on their bypass LSPs.

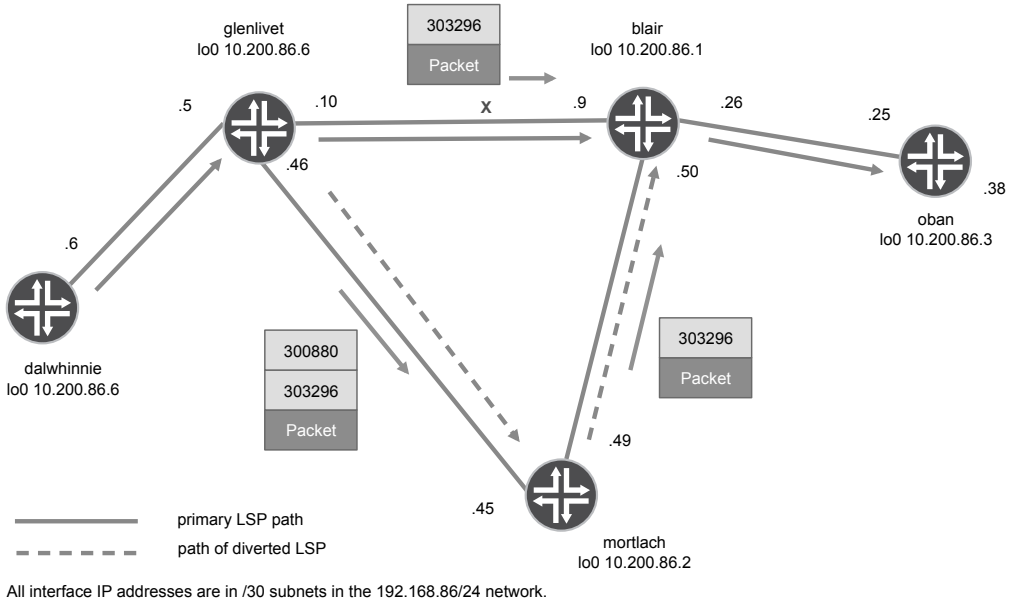



Figure 1.6 Link-protection Label Operations

Configure link protection by including the `link-protection` keyword within the MPLS label-switched-path *and* within each interface at the `[edit protocols rsvp]` hierarchy level. Without this configuration at both levels, the LSP will not be link-protected: It is important to understand why link protection must be configured in those two areas. The `link-protection` keyword within the LSP configuration directs the LSP's ingress and transit routers to protect that LSP with a next-hop bypass if one is available. The link-protection configuration within the interface hierarchy in the `[edit protocols rsvp]` stanza, on the other hand, notifies RSVP on a router that it needs to signal a bypass LSP if a link-protected LSP transits that specific RSVP interface. Another way to consider the difference between the two is that the LSP configuration (found only on the ingress router) directs all the LSRs in the LSP's path to signal their bypasses, something each can do only if its egress RSVP interface is configured for link protection.

```
[edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    link-protection;
}
label-switched-path dalwhinnie-to-oban-lo-priority {
    to 10.200.86.3;
    link-protection;
}
path via-tormore;
interface ge-0/0/2.0;
interface ge-0/0/3.0;

[edit protocols rsvp]
ps@dalwhinnie# show
interface ge-0/0/2.0 {
    link-protection;
}
interface ge-0/0/3.0 {
    link-protection;
}
```

One drawback to link protection is that the link protection capability must be configured on the desired RSVP interfaces on each transit router along the LSP, resulting in a slightly larger router configuration for each router.

TIP When using link protection, it is a best practice to enable the link protection capability on every RSVP interface on each router in the RSVP domain in order to ensure that link protection for an RSVP LSP is available no matter which routers or core-facing interfaces the LSP may transit at a given point in time.

## Node-link Protection

Node-link protection builds on the failover protection offered by link protection. Node-link protection provides for the signaling of two types of failover LSPs: the *next-hop bypass LSP* discussed in the previous section and the *next-next-hop bypass LSP*. The next-next-hop bypass LSP's purpose is to offer an alternate path toward the next-hop router's next-hop router for the traffic on a protected LSP in order to protect the LSP in the event that the LSP's next-hop router experiences a down link or a catastrophic hardware or software failure. In other words, the next-next-hop bypass LSP provides an alternate path to the next-hop's next-hop router. If network topology is not rich enough to provide for a next-next-hop bypass LSP, or if a router is the penultimate router in the LSP, then a router signals a *next-hop bypass LSP* instead. If the network topology does not support a next-hop bypass LSP, then that bypass LSP is not created, but the router still supports the original LSP. Figure 1.7 shows a scenario for setup of a next-next-hop bypass LSP created by node-link protection.

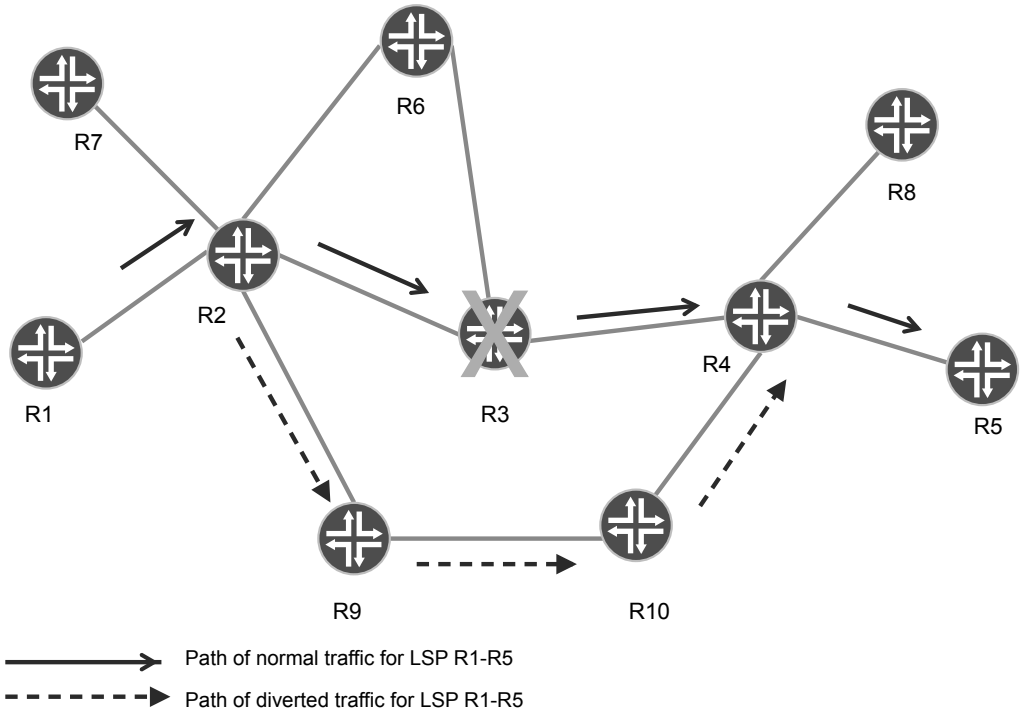



Figure 1.7 R1-R5 LSP and Next-next-hop Bypass LSP Spawned by Node-link Protection

Configure node-link protection by including the `node-link-protection` keyword within each MPLS LSP where this protection is desired. Additionally, link protection

must also be configured for each RSVP interface that the node-link protected LSP may traverse. In this case, when the local router sees an LSP requesting node-link protection, it attempts to signal a bypass LSP to the next next-hop router for the LSP; if it cannot do so then it signals a next-hop bypass LSP. As is the case with link-protection, it is a best practice to configure link-protection under every interface configured for RSVP on each router in the RSVP domain if node-link protection is enabled. This ensures that any core-facing interface that the LSP may traverse is able to provide the node-link protection. Here the `node-link-protection` keyword is included in the LSP configuration:

```
ps@dalwhinnie> show configuration protocols mpls
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    node-link-protection;
}
interface ge-0/0/2.0;
interface ge-0/0/3.0;

ps@dalwhinnie> show configuration protocols rsvp
interface ge-0/0/2.0 {
    link-protection;
}
interface ge-0/0/3.0 {
    link-protection;
}
```

Each transit router along the *dalwhinnie-to-oban* LSP signals a next-next-hop bypass LSP for node protection if the architecture supports it. If the architecture cannot support that type of LSP, then the router will attempt to signal a next-hop bypass LSP for link-protection. If the architecture cannot support a next-hop bypass LSP, then each router in the path will simply support the LSP itself. Note that the router does *not* signal both a next-hop and next-next-hop LSP to protect a node-link protected LSP – it signals one or the other based on what the architecture supports.

WARNING Much of the official Juniper documentation on node-link protection suggests that each ingress and transit router in an LSP's path signals *both* a next-next-hop LSP AND a next-hop LSP. However, this book's lab results show that each of these routers signals one or the other: the router signals a next-next-hop bypass LSP for protection if the network architecture supports it; if it does not, then, and only then, will the router signal a next-hop bypass LSP for a node-link protected LSP's protection.

The *dalwhinnie-to-oban* LSP, configured for node-link protection, is up:

```
ps@dalwhinnie> show mpls lsp detail
Ingress LSP: 1 sessions

10.200.86.3
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-oban
  ActivePath:  (primary)
  Node/Link protection desired
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 3)
 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt
```

```
20=Node-ID):
        10.200.86.6(flag=0x29) 192.168.86.5(flag=9 Label=300416)
10.200.86.1(flag=0x21) 192.168.86.9(flag=1 Label=300400) 10.200.86.3(flag=0x20)
192.168.86.25(Label=3)
```

The dalwhinnie-to-oban LSP traverses glenlivet and blair, terminating on oban (see Figure 1.3). It is configured such that each router in the path, minus the egress router, attempts to signal a next-next-hop or next-hop bypass type protection LSP.

Dalwhinnie signals a next-next-hop bypass LSP for protection:

```
ps@dalwhinnie> show mpls lsp bypass ingress
Ingress LSP: 2 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.1    10.200.86.5    Up      0  1 SE      -   300384 Bypass->192.168.86.5->192.168.86.9
Total 1 displayed, Up 1, Down 0

ps@dalwhinnie> show mpls lsp bypass name Bypass->192.168.86.5->192.168.86.9 detail
Ingress LSP: 2 sessions

10.200.86.1
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass->192.168.86.5->192.168.86.9
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300848
  Resv style: 1 SE, Label in: -, Label out: 300848
  Time left:   -, Since: Wed Dec  8 05:54:30 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 1845 protocol 0
  Type: Bypass LSP
    Number of data route tunnel through: 1
    Number of RSVP session tunnel through: 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.29 (ge-0/0/3.0) 4 pkts
  RESV rcvfrom: 192.168.86.29 (ge-0/0/3.0) 4 pkts
  Explct route: 192.168.86.29 192.168.86.1 192.168.86.13 192.168.86.18
  Record route: <self> 192.168.86.29 192.168.86.1 192.168.86.13 192.168.86.18
Total 1 displayed, Up 1, Down 0

Egress LSP: 3 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 3 sessions
Total 0 displayed, Up 0, Down 0
```

Dalwhinnie's next-next-hop LSP is named *Bypass->192.168.86.5->192.168.86.9*, and it traverses lagavulin, macduff, and talisker, egressing on blair. It completely bypasses glenlivet, offering protection if the entire glenlivet node or the link to the glenlivet node fails. Notice that the router does not create a separate next-hop type bypass LSP to protect the link to glenlivet – the next-next-hop LSP is offering protection for both the next-hop link and node.

Finally, look at the *dalwhinnie-to-oban* LSP and its protection LSP at blair, the LSP's penultimate router:

```
ps@blair> show mpls lsp transit name dalwhinnie-to-oban detail
Transit LSP: 12 sessions

10.200.86.3
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 1
  LSPname: dalwhinnie-to-oban, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
```

```
  Recovery label received: -, Recovery label sent: 3
  Resv style: 1 SE, Label in: 303744, Label out: 3
  Time left:  144, Since: Wed Dec  8 07:21:19 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 3 receiver 1843 protocol 0
  Node/Link protection desired
  Type: Link protected LSP
  PATH rcvfrom: 192.168.86.10 (ge-0/0/1.0) 41 pkts
  Adspec: received MTU 1500 sent MTU 1500
  PATH sentto: 192.168.86.25 (ge-6/0/0.0) 43 pkts
  RESV rcvfrom: 192.168.86.25 (ge-6/0/0.0) 41 pkts
  Explct route: 192.168.86.25
  Record route: 192.168.86.6 192.168.86.10 <self> 10.200.86.3 (node-id) 192.168.86.25
Total 1 displayed, Up 1, Down 0

ps@blair> show mpls lsp bypass ingress
Ingress LSP: 2 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.3    10.200.86.1    Up      0  1 SE      -   301360 Bypass->192.168.86.25
10.200.86.4    10.200.86.1    Up      0  1 SE      -   301168 Bypass->192.168.86.17
Total 2 displayed, Up 2, Down 0

ps@blair>

ps@blair> show mpls lsp bypass name Bypass->192.168.86.25 detail
Ingress LSP: 2 sessions

10.200.86.3
  From: 10.200.86.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass->192.168.86.25
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 301360
  Resv style: 1 SE, Label in: -, Label out: 301360
  Time left:    -, Since: Wed Dec  8 05:21:15 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 49268 protocol 0
  Type: Bypass LSP
    Number of data route tunnel through: 2
    Number of RSVP session tunnel through: 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.17 (ge-0/0/3.0) 185 pkts
  RESV rcvfrom: 192.168.86.17 (ge-0/0/3.0) 185 pkts
  Explct route: 192.168.86.17 192.168.86.33 192.168.86.38
  Record route: <self> 192.168.86.17 192.168.86.33 192.168.86.38
Total 1 displayed, Up 1, Down 0

Egress LSP: 3 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 12 sessions
Total 0 displayed, Up 0, Down 0

ps@blair>
```

Blair needs to protect the link to oban, where the LSP terminates. Although blair is aware that the LSP requests node/link protection, it cannot offer node protection since it is the penultimate router (no sense in bypassing the LSP's egress node). Instead of a next-next-hop bypass LSP, blair signals the next-hop bypass LSP *Bypass->192.168.86.25* to protect the link to the LSPs egress router.

There are some differences in performance between the next-hop bypass LSP and

the next-next-hop bypass LSP. As explained previously, link protection relies on the router's hardware to detect a link failure, and so the PLR router can quickly detect the break and switch the protected traffic to the next-hop LSP. Node-protection, on the other hand, relies on the receipt of hello messages from the neighboring router to determine if that router still exists. The time it takes for the PLR router to switch traffic onto a next-next-hop LSP is dependent on the frequency of hello messages from the neighboring router and the time it takes the PLR router to detect the absence of those messages. Once the failure is detected, however, the PLR can quickly switch the traffic to the next-next-hop bypass LSP. Because of this difference between how next-hop and next-next-hop LSPs detect failures, when using node-protection it is reasonable to expect slightly longer failover times for your next-next-hop LSPs as compared to the failover times for a next-hop LSP. This difference in failover time may be an important factor in determining whether to employ link-node-protection or link-protection. Another closely related factor to consider is the likelihood of an entire node failing, especially if redundant routing-engines and power supplies are available for your specific router model.

Traffic traveling on the next-hop bypass LSP, or next-next-hop bypass LSP, is a short term fix. When the PLR switches traffic to the bypass LSP, it also signals back to the LSP's ingress router. The ingress router then attempts to signal the LSP again.

Like the next-hop bypass LSP, the next-next-hop bypass LSP is able to protect traffic for multiple LSPs, making it very scalable. Like link protection, node-link protection capability must be configured on each transit router's RSVP interfaces along the LSP.

## Fast Reroute

Fast Reroute (FRR) is a Juniper proprietary implementation of LSP failover protection. Fast reroute instructs each router in the LSP (except for the egress router) to signal a single alternate path toward the LSP destination that avoids the link to the next downstream node *and* to the next downstream node itself. These alternate paths are called *detours*. If the network topology is not rich enough to support a specific fast reroute detour, then that detour is not created.

When a failure on the fast-reroute protected LSP occurs, the detour may not be the optimal path to the LSP's destination. As a permanent fix, the ingress LSR signals a new path in a *make-before-break* fashion, and gracefully moves the traffic from the old, broken LSP path to the new, optimal LSP path. Factors that determine how much traffic is lost during an LSP break include how long it takes the PLR router to detect the link break (the router can detect a SONET link failure quicker than a GigE failure, and detects link failures faster than a node failure, for example) and how long it takes to move the traffic on to the pre-signaled detour.

Unlike link-protection and node-link protection, when a failure along the LSP path occurs, there is no bypass label added to the label stack: the PLR router simply *swaps* the top label and changes the packet's egress interface; the height of the label stack does not change. Another important difference between FRR and link-protection and node-link-protection is that each FRR detour that each router along the LSP generates can only be used to protect traffic for one specific LSP. Within a single LSP's path containing *N* routers, it is possible that fast reroute will generate *N-1* detours for that LSP (all routers but the egress node can generate a detour), each detour is only capable of protecting the one specific LSP. Multiply this by the total number of LSPs that a given network would need and it is quickly apparent that fast reroute does not scale well.

Fast reroute does implement a couple of measures meant to reduce the scalability impact. A fast reroute detour will attempt to merge back in to the original LSP path in as few hops as possible. This minimizes the amount of router overhead required to maintain each detour. Additionally, if topology constraints prevent the detour from quickly merging back into the LSP path, then one detour can merge with another detour for the same LSP. Figure 1.8 shows an example of this scenario. R1 signals a detour that goes around the R1-R2 link and around R2 itself, but wants the detour to merge back on the LSP path in as few hops as possible. R2 signals a detour that goes around the R2-R3 link, and R3 itself, and also wants that detour to merge back on to the LSP path in as few hops as possible. Both detours can share the R9-R10 and R10-R4 links for the purpose of merging back to the original LSP path in as few hops as possible, if necessary.

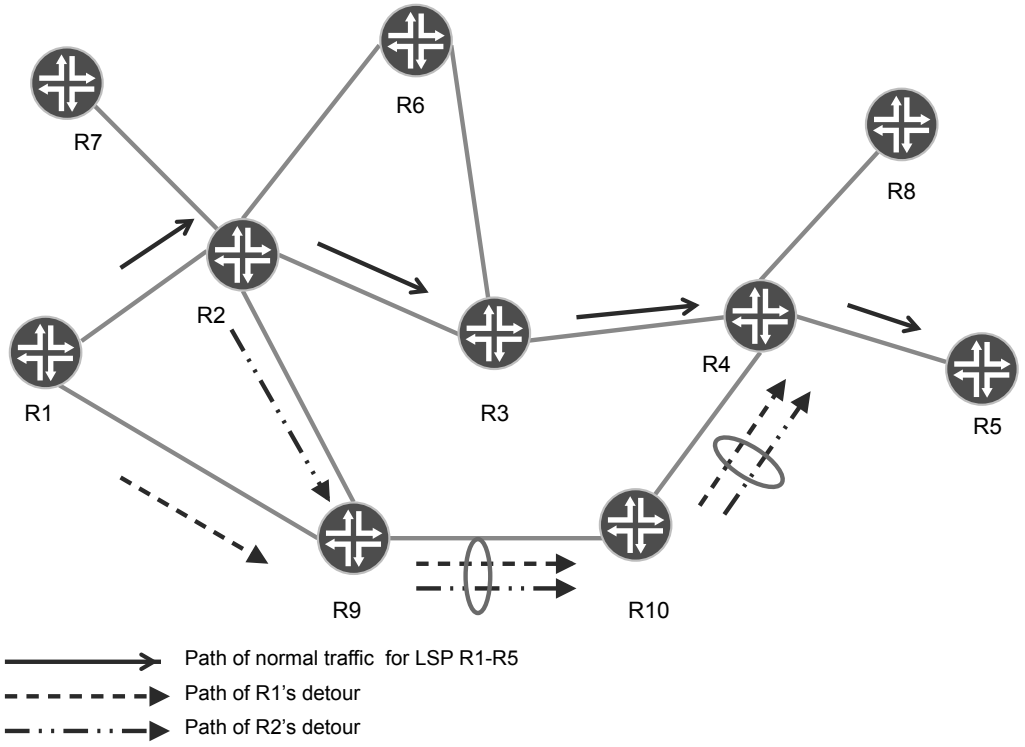



Figure 1.8 R1 and R2 Sharing Fast Reroute Detour Links

Enable an LSP for fast reroute by configuring the `fast-reroute` keyword within the specific label-switched-path where it is desired. Unlike link-protection and node-link-protection, fast reroute only needs to be configured on the LSP ingress router. When the LSP is set up, the ingress router notifies all downstream routers that fast reroute is desired; the downstream routers then attempt to set up the detours. If a specific detour from a router cannot be set up for whatever reason, that router continues to support the LSP.

The code snippet below from dalwhinnie shows fast-reroute configured for the *dalwhinnie-to-oban* LSP. Adding the `fast-reroute` keyword on the ingress router is the only configuration necessary to protect this LSP.

```
[edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
```

```
    to 10.200.86.3;
    fast-reroute;
}
interface ge-0/0/2.0;
interface ge-0/0/3.0;

[edit protocols mpls]
ps@dalwhinnie#
```

NOTE If a fast-reroute protected LSP has any *link-coloring* restrictions, the FRR detours also inherit those restrictions. Link-coloring is covered in the *Traffic Engineering* section of this chapter.

To verify the status of the fast-reroute RSVP LSP, the LSP path, and the status of the detour from the ingress router, use the `show rsvp session detail` operational command for the specific LSP. The following sample output shows the IP addresses of the record route that the LSP is taking through the network, notes that the fast reroute detour path is up, and displays the detour path's record route:

```
ps@dalwhinnie> show rsvp session lsp name dalwhinnie-to-oban detail
Ingress RSVP: 1 sessions

10.200.86.3
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 0
  LSPname: dalwhinnie-to-oban, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300432
  Resv style: 1 FF, Label in: -, Label out: 300432
  Time left:   -, Since: Wed Jul 21 03:36:04 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 1773 protocol 0
  FastReroute desired
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.5 (ge-0/0/2.0) 37 pkts
  RESV rcvfrom: 192.168.86.5 (ge-0/0/2.0) 34 pkts
  Explct route: 192.168.86.5 192.168.86.9 192.168.86.25
  Record route: <self> 192.168.86.5 192.168.86.9 192.168.86.25
    Detour is Up
    Detour Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
    Detour adspec: sent MTU 1500
    Path MTU: received 1500
    Detour PATH sentto: 192.168.86.29 (ge-0/0/3.0) 34 pkts
    Detour RESV rcvfrom: 192.168.86.29 (ge-0/0/3.0) 32 pkts
    Detour Explct route: 192.168.86.29 192.168.86.1 192.168.86.13
    192.168.86.33 192.168.86.38
    Detour Record route: <self> 192.168.86.29 192.168.86.1 192.168.86.13
    192.168.86.33 192.168.86.38
    Detour Label out: 300400
Total 1 displayed, Up 1, Down 0
```

The dalwhinnie-to-oban LSP traverses glenlivet and blair and terminates on oban. The FRR detour is up and traverses lagavulin, macduff, talisker, and tormore and terminates on oban (refer back to Figure 1.3).

Any router along an LSP's record route can also display the status and path of a detour LSP originating from the router. In the output below, the transit router glenlivet has signaled a detour to oban, the LSP's destination.

```
ps@glenlivet> show rsvp session transit detail lsp name dalwhinnie-to-oban
Transit RSVP: 1 sessions
```

```
10.200.86.3
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 1
  LSPname: dalwhinnie-to-oban, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300416
  Resv style: 1 FF, Label in: 300432, Label out: 300416
  Time left:  130, Since: Wed Jul 21 03:34:58 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 1773 protocol 0
  FastReroute desired
  PATH rcvfrom: 192.168.86.6 (ge-0/0/2.0) 44 pkts
  Adspec: received MTU 1500 sent MTU 1500
  PATH sentto: 192.168.86.9 (ge-0/0/1.0) 42 pkts
  RESV rcvfrom: 192.168.86.9 (ge-0/0/1.0) 40 pkts
  Explct route: 192.168.86.9 192.168.86.25
  Record route: 192.168.86.6 <self> 192.168.86.9 192.168.86.25
    Detour is Up
    Detour Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
    Detour adspec: received MTU 1500 sent MTU 1500
    Path MTU: received 1500
    Detour PATH sentto: 192.168.86.45 (ge-0/0/3.0) 40 pkts
    Detour RESV rcvfrom: 192.168.86.45 (ge-0/0/3.0) 37 pkts
    Detour Explct route: 192.168.86.45 192.168.86.42 192.168.86.13
    192.168.86.33 192.168.86.38
    Detour Record route: 192.168.86.6 <self> 192.168.86.45 192.168.86.42
    192.168.86.13 192.168.86.33 192.168.86.38
    Detour Label out: 300192
Total 1 displayed, Up 1, Down 0

ps@glenlivet>
```

The detour from glenlivet, protecting the LSP *dalwhinnie-to-oban*, traverses mortlach, macduff, talisker, and tormore, and terminates on oban.

For smaller networks, fast-reroute offers a simple-to-configure and reliable method of failover, but may have scalability problems as the network grows. Since FRR is not highly scalable and is Juniper proprietary, it is recommended that this failover method not be used unless you are reasonably sure that the RSVP domain will remain smaller in scale and comprised only of Juniper routers.

## Secondary LSP

Another option available to protect an LSP against a failure is use of a *secondary* LSP. One or more secondary paths can be configured for an LSP. Configuration of a secondary LSP causes two or more LSP paths to be set up: a primary path to be used initially and one or more secondary paths that can be used in the event that a failure occurs in the primary path. Like the primary path, the secondary path(s) terminate on the LSP's egress router. The Junos router waits until the primary path is established and then searches the traffic-engineering database (TED) for suitable secondary paths. If no completely diverse path exists the secondary LSP may contain common links with the primary LSP. In the example shown below, the *dalwhinnie-to-oban* LSP has a primary and secondary path configured. The `standby` keyword instructs the LSP to signal the secondary path to remain up at all times in order to cover in case of a failover. Without `standby` configured on the secondary path, the LSP waits until the primary path fails before establishing the secondary path, greatly increasing the time for failover. This configuration snippet from dalwhinnie shows the *dalwhinnie-to-oban* LSP configured to immediately signal a secondary LSP:

```
[edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    primary via-blair;
    secondary via-tormore {
        standby;
    }
}
path via-tormore {
    10.200.86.9 loose;
}
path via-blair {
    192.168.86.5 strict;
    192.168.86.9 strict;
    192.168.86.25 strict;
}
interface ge-0/0/2.0;
interface ge-0/0/3.0;
```

This configuration generates an LSP with a strict primary path following glenlivet, blair, and oban and a pre-signaled loose secondary path that must contain tormore. The use of strict and loose hops in label-switched-path paths is optional. However, specifying strict and loose hops gives network engineers options for traffic-engineering. (The concept of, and tools used in, traffic engineering are covered later in this chapter.)

NOTE Hops in a path can be defined as *loose* or *strict*. An LSP path must traverse all configured hops in the order in which they are configured. At any given router in the LSP path, the next strict hop must be directly connected to the router, while a loose hop can be any hop reachable in the network from that router.

The following operational output from dalwhinnie shows information about the *dalwhinnie-to-oban* LSP, including the record route object (RRO) for both the primary and secondary (standby) paths. The RRO is the ordered list of IP addresses for each ingress interface on each node in the LSP.

```
ps@dalwhinnie> show mpls lsp name dalwhinnie-to-oban detail
Ingress LSP: 2 sessions

10.200.86.3
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair       State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 3)
 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.5 192.168.86.9 192.168.86.25
  Standby   via-tormore     State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 5)
 192.168.86.29 S 192.168.86.1 S 192.168.86.13 S 192.168.86.33 S 192.168.86.38 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.29 192.168.86.1 192.168.86.13 192.168.86.33 192.168.86.38
Total 1 displayed, Up 1, Down 0

Egress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

```
Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

The *dalwhinnie-to-oban* LSP's primary path is *via-blair*, hitting the glenlivet, blair, and oban routers. The standby path is *via-tormore* and is currently up; this path traverses lagavulin, macduff, talisker, and tormore, terminating on oban. Figure 1.9 illustrates these paths.

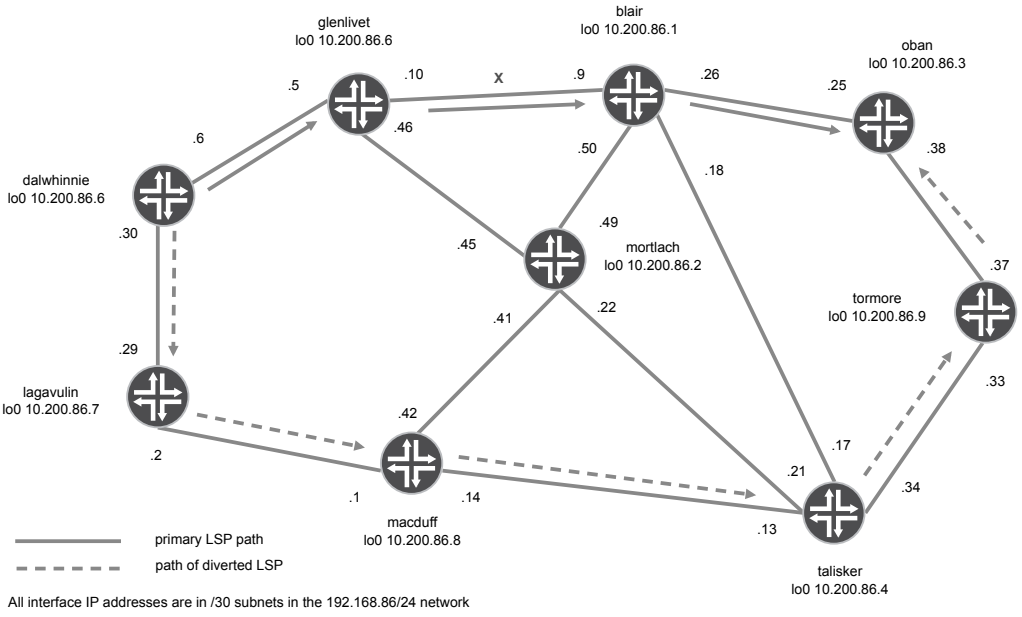



Figure 1.9 Primary and Secondary Paths for LSP Dalwhinnie-to-oban

WARNING Even properly planned failover scenarios using link-protection, node-link protection, or secondary LSPs may not tolerate multiple network link or router failures.

Designating an LSP path as *primary* means that the primary path must be used if it is available in the network. In other words, a primary path is revertive: if an LSP's primary path goes down and traffic switches to the secondary path, the traffic switches back to the primary path if and when it becomes available again. In order to prevent this revertive behavior, instead of using a primary and secondary path, two or more secondary paths can be configured and put in standby (no primary is configured). The LSP signals the secondary paths in the order in which they are configured under the specific LSP. If the active path fails, the traffic fails over to the other secondary path. If the previously active path comes back up, however, traffic will still remain on the other secondary path until that path fails. Use of the standby keyword for both secondary LSP paths ensures that the LSP will attempt to signal both paths and keep both paths up, no matter which path is active at any given time. The configuration from dalwhinnie shows the *dalwhinnie-to-tormore* LSP has two secondary paths configured:

```
[edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
   to 10.200.86.3;
   primary via-blair;
   secondary via-tormore {
```

```
            standby;
        }
    }
    label-switched-path dalwhinnie-to-tormore {
        to 10.200.86.9;
        secondary via-talisker {
            standby;
        }
        secondary alt-path {
            standby;
        }
    }
    path via-tormore {
        10.200.86.9 loose;
    }
    path via-talisker {
        10.200.86.4 loose;
    }
    path via-blair {
        192.168.86.5 strict;
        192.168.86.9 strict;
        192.168.86.25 strict;
    }
    path alt-path;
    interface ge-0/0/2.0;
    interface ge-0/0/3.0;
```

The path *alt-path*, in this case, does not have any hops defined, so the LSP will simply attempt to signal the best path through the network. The *via-talisker* path has one loose hop configured. A detailed look at the *dalwhinnie-to-tormore* LSP shows both secondary LSP paths up:

```
[edit protocols mpls]
ps@dalwhinnie# run show mpls lsp name dalwhinnie-to-tormore detail
Ingress LSP: 2 sessions

10.200.86.9
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-tormore
  ActivePath: via-talisker (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Standby   via-talisker    State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
 192.168.86.5 S 192.168.86.9 S 192.168.86.17 S 192.168.86.33 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.5 192.168.86.9 192.168.86.17 192.168.86.33
  Standby   alt-path       State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
 192.168.86.5 S 192.168.86.9 S 192.168.86.17 S 192.168.86.33 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.5 192.168.86.9 192.168.86.17 192.168.86.33
Total 1 displayed, Up 1, Down 0

Egress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

Careful analysis of this output shows an immediate problem: both secondary paths are taking the same route through the network. This is a possible outcome if paths do not contain enough strict or loose hops to ensure that each path protecting an LSP takes a unique route through the network. When using secondary paths to provide failover, it is very important to understand how network topology, IGP link costs, and strict and loose hops for a path can affect path diversity.

Finally, it is useful to understand that a single defined path name can have different record routes if it is applied to different LSPs. In this next configuration, *alt-path* is designated as the secondary path for two LSPs:

```
ps@dalwhinnie> show configuration protocols mpls
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    primary via-blair;
    secondary alt-path {
        standby;
    }
}
label-switched-path dalwhinnie-to-tormore {
    to 10.200.86.9;
    secondary via-talisker {
        standby;
    }
    secondary alt-path {
        standby;
    }
}
. . . <snipped for brevity> . . .
 path alt-path;
```

Both of the *dalwhinnie-to-oban* and *dalwhinnie-to-tormore* LSPs show *alt-path* as a secondary path for the LSP.

The output below shows that the path *alt-path* is taking two separate paths through the network, depending on which LSP it is configured to protect:

```
ps@dalwhinnie> show mpls lsp detail
Ingress LSP: 2 sessions

10.200.86.3
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair       State: Up
     . . .
     . . . <snipped for brevity> . . .
     . . .
  Standby   alt-path         State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 5)
 192.168.86.29 S 192.168.86.1 S 192.168.86.13 S 192.168.86.33 S 192.168.86.38 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.29 192.168.86.1 192.168.86.13 192.168.86.33 192.168.86.38

10.200.86.9
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-tormore
  ActivePath: via-talisker (secondary) dalwhinnie-to-oban
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
```

```
*Standby   via-talisker    State: Up
    . . .
    . . . <snipped for brevity> . . .
    . . .
 Standby   alt-path         State: Up
   Priorities: 7 0
   SmartOptimizeTimer: 180
   Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
192.168.86.5 S 192.168.86.9 S 192.168.86.17 S 192.168.86.33 S
   Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
        192.168.86.5 192.168.86.9 192.168.86.17 192.168.86.33
```

When *alt-path* protects the *dalwhinnie-to-oban* LSP, it takes the lagavulin-macduff-talisker-tormore-oban path; when it protects *dalwhinnie-to-tormore*, *alt-path* takes the glenlivet-blair-talisker-tormore path.

In a failover scenario, secondary LSPs are typically the slowest alternative and they are not always reliable. This is because news of the link break or node failure must travel upstream to the ingress router via a RESV TEAR message, which is not reliable. Only then, if the ingress router receives the message, will the ingress router move traffic from the active to the standby path. If the link break results in traffic saturation of other links, then the RESV TEAR message may be dropped before it reaches the ingress router. At this point, the ingress router must then wait for IGP to reconverge (which may be many seconds) before knowing about the link or node failure; during this wait time, traffic on the LSP is black-holed. Compare the secondary LSP failover method to link protection, node-link protection, and fast reroute, where traffic is redirected immediately at the closest router to the network failure (and thus the first router to learn of the failure), and it is clear why using secondary LSPs would result in a longer time to switch traffic.

## Loop-Free Alternates in IGPs

Another failover option we'll discuss briefly is the use of loop-free alternate paths (LFA paths) in IGPs. Because OSPF and ISIS are link-state protocols, they build a routing-table based on network topology, giving them visibility of the entire OSPF/OSPF3 area or ISIS level. In Junos 9.5 and beyond for ISIS, and 10.0 and beyond for OSPF, these protocols can be configured to provide link protection or node-link protection for a given egress interface. After its initial SPF run, the IGP (OSPF or ISIS) on each router removes either the egress link (for link protection) or the next-hop node (for node-link protection) from the topology and reruns the Dijkstra algorithm. Junos pre-installs these alternate paths into the packet-forwarding engine (PFE), allowing the PFE to begin forwarding traffic around a failed link or node before receiving an updated forwarding information base (FIB) from the routing-engine. LFA IGP paths provide protection for each route that egresses the configured interface. For a network running LDP as the MPLS protocol, LFA provides fast-failover protection for LDP LSPs that egress the configured interface. This capability may make LDP a more attractive option for a network that requires MPLS services (L2VPN, L3VPN, and VPLS) and simple fast failover but does not require the traffic-engineering (TE) capabilities that RSVP provides. This chapter covers TE in the next section.

The following configuration shows how to configure LFA link protection or node-link protection in OSPF and OSPF3.

```
[edit protocols]
ps@dalwhinnie# show ospf
```

```
area 0.0.0.0 {
   interface ge-0/0/0.0 {
      link-protection;
   }
   interface ge-0/0/1.0 {
      node-link-protection;
   }
}

[edit protocols]
ps@dalwhinnie# show ospf3
area 0.0.0.0 {
   interface ge-0/0/0.0 {
      link-protection;
   }
   interface ge-0/0/1.0 {
      node-link-protection;
   }
}
```

And here is how LFA looks in an ISIS configuration:

```
[edit protocols]
ps@dalwhinnie# show isis
interface fe-0/0/3.0 {
   link-protection;
   level 1 disable;
}
interface fe-0/0/4.0 {
   node-link-protection;
   level 1 disable;
}
```

Note that an interface can provide link protection or node-link protection, but cannot provide both. By default, all OSPF or ISIS interfaces in the master instance or a routing-instance are eligible to act as backup interfaces for link or node-link protected interfaces. To disable an interface from providing backup capability, include the *no-eligible-backup* keyphrase under the specific interface. In the example below, interface ge-1/0/0.0 is excluded from providing an alternate egress for LFA protected interfaces.

```
[edit protocols]
tim@srx100# show ospf area 0 interface ge-1/0/0.0
no-eligible-backup;
```

TIP When configuring link protection, it is also necessary to configure a per-packet load-balancing routing policy to ensure that the IGP installs all the next-hops for a given route into the routing table. Below is a very basic per-packet load-balancing policy; apply the policy at the [edit routing-options forwarding-table] level in the hierarchy.

```
 [edit]
ps@dalwhinnie# show policy-options
policy-statement basic-load-balancing-policy {
   then {
      load-balance per-packet;
   }
}

[edit]
ps@dalwhinnie# show routing-options
```

```
 . . .
 . . .
router-id 10.200.86.5;
autonomous-system 7176;
forwarding-table {
   export basic-load-balancing-policy;
}
. . .
. . .
```

CAUTION LFA faces some limitations in certain topologies. That discussion is outside the scope of this book.

## Traffic Engineering

Traffic engineering (TE) is a method of optimizing network resources and performance by regulating traffic across the network backbone. Traffic engineering (typically) involves classification of traffic based on specific criteria and then specifying which network paths that traffic may use to traverse the backbone. In this way, an organization can help ensure that time-sensitive and critical traffic can take the most expedient route, while traffic for non-real time and low-priority traffic such as email can take a different, perhaps less direct, route. Enforcing this type of segmentation can help ensure that no specific router or link is under-utilized or over-utilized. MPLS offers several traffic-engineering tools to allow network architects and engineers to make the best use of their network resources.

### LSP Metrics

It is possible to manually assign a static metric to an RSVP LSP, in the same fashion that a metric can be assigned to an IGP interface. Once configured, the LSP cost is fixed and cannot change, regardless of the underlying IGP metrics of the interfaces along the LSP's path. Keep this in mind, because even if the LSP activates a failover method such as link protection and is then re-signaled, the LSP metric remains fixed at the configured value. In this fashion, if multiple LSPs exist to a specific destination, it is possible to force all traffic onto one LSP or to load-share traffic between LSPs.

Configure an LSP's metric at the specific LSP's hierarchy, as shown below:

```
[edit protocols mpls]
ps@dalwhinnie# show label-switched-path dalwhinnie-to-oban
to 10.200.86.3;
metric 20;
link-protection;
```

### Link Coloring

Link coloring allows network engineers and architects to mark certain network links with one or more of 32 possible values (0-31). Once these links are marked, any specific LSP can be configured to travel only on certain links and/or to never travel on other links. Junos allows this type of link marking via *admin-groups*. Admin-groups match a user-defined group name to a numerical value 0-31 – the group name is simply a user-friendly mnemonic that adds meaning instead of a numerical value that provides little context.

Examine Figure 1.10. It shows the network with the links dalwhinnie-glenlivet, glenlivet-blair, and blair-oban colored (named) *gold*.

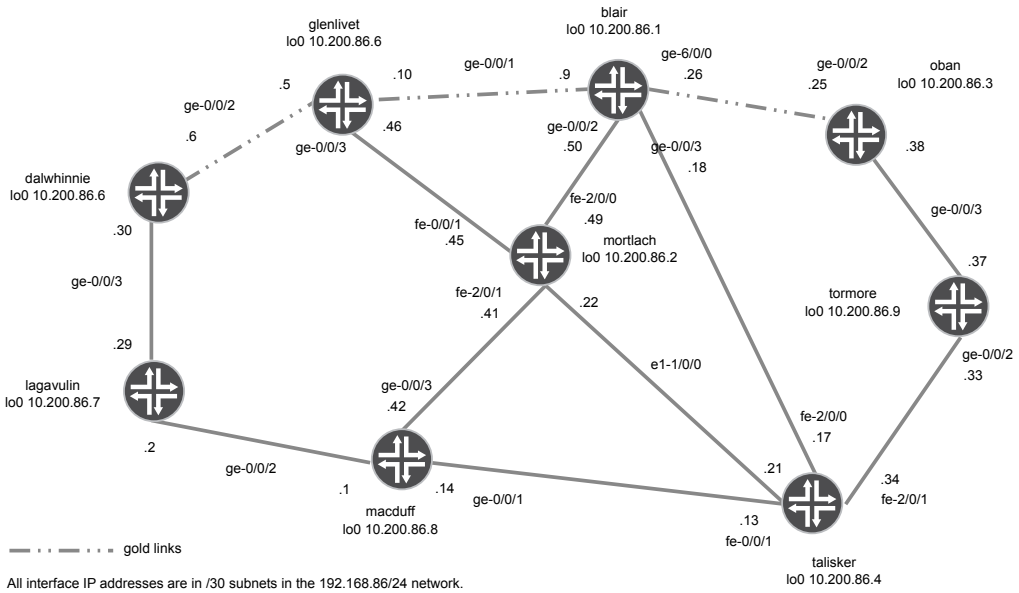



Figure 1.10 Link Coloring

The configuration example below shows the *gold* administrative group and two configured LSPs. Interface ge-0/0/2.0 on dalwhinnie is configured in the *gold* admin-group. Additionally, the label-switched-path *dalwhinnie-to-oban-lo-priority* is configured to avoid *gold* links.

Note that the gold link coloring is also configured on the specific interfaces on glenlivet (ge-0/0/1.0) and blair (ge-6/0/0.0) under the `[edit protocols mpls]` hierarchy (not shown). This has two effects on the LSPs:

- LSP dalwhinnie-to-oban is required to take gold links through the network to reach oban. Any bypass LSPs protecting this link, however, would not be required to contain gold links in their path. This is because a bypass LSP can be used to protect multiple LSPs that may have many different constraints. The fast reroute detours for an LSP protected by FRR, on the other hand, would inherit the primary LSP's constraints and be required to take gold links in our example.
- LSP dalwhinnie-to-oban-lo-priority is explicitly forbidden to take any gold colored links. Note, however, that if this LSP is configured for link protection, then the bypass LSPs protecting this LSP may take gold links, unless expressly forbidden to do so. Primary and secondary paths and bypass LSPs can be configured with admin-group restrictions as well. Keep in mind that if this LSP is configured for fast reroute, instead of a link-protection bypass LSP, then the FRR detour *would* be forbidden to take gold links.

NOTE Because LSPs are unidirectional, if the network engineer desires an equivalent *gold* path in the opposite direction, from oban to dalwhinnie, then she must provision an LSP on oban to dalwhinnie and then add mpls interfaces ge-0/0/2.0 on oban, ge-0/0/1.0 on blair, and ge-0/0/1.0 on glenlivet to the *gold* admin-group.

The MPLS configuration with gold link coloring on dalwhinnie would be:

```
ps@dalwhinnie> show configuration protocols mpls
admin-groups {
    gold 1;
}
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    admin-group include-any gold;
    link-protection;
}
label-switched-path dalwhinnie-to-oban-lo-priority {
    to 10.200.86.3;
    admin-group exclude gold;
    link-protection;
}
path via-tormore;
interface ge-0/0/2.0 {
    admin-group gold;
}
interface ge-0/0/3.0;

ps@dalwhinnie> show configuration protocols rsvp
interface ge-0/0/2.0 {
    link-protection;
}
interface ge-0/0/3.0 {
    link-protection;
}
```

The ge-0/0/2.0 interface is added to the *gold* admin-group. The *dalwhinnie-to-oban* LSP is required to take *gold* links, due to the *include-any gold* configuration. Since ge-0/0/2.0 is dalwhinnie's only *gold* link, the *dalwhinnie-to-oban* LSP must egress the router through that interface. Note that if there is not a complete path of gold links from dalwhinnie to oban, then the *dalwhinnie-to-oban* LSP will fail to come up.

Here is a configuration showing primary and secondary LSP paths with admin-group restrictions:

```
[edit protocols mpls]
ps@dalwhinnie# show
admin-groups {
    gold 1;
}
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    primary via-tormore {
        admin-group include-all gold;
    }
    secondary alt-path {
        admin-group include-all gold;
    }
}
interface ge-0/0/2.0 {
    admin-group gold;
}
```

And a bypass LSP configured for a requirement to include gold links. Note that this is configured within the [protocols rsvp] stanza:

```
[edit protocols rsvp]
ps@dalwhinnie# show
```

```
interface ge-0/0/2.0 {
   link-protection {
      admin-group include-any gold;
   }
}
```

When configuring admin-group restrictions, the options are *exclude*, *include-all*, or *include-any* -

```
[edit protocols mpls label-switched-path dalwhinnie-to-oban]
ps@dalwhinnie# set primary via-tormore admin-group ?
Possible completions:
 . . .
+ exclude            Groups, all of which must be absent
+ include-all         Groups, all of which must be present
+ include-any         Groups, one or more of which must be present
```

NOTE When there are no admin-group constraints on an LSP, then that LSP is allowed to consider possible paths through the network regardless of any link coloring configured on any router interfaces.

Link-coloring is an effective way to reserve specific links for high priority traffic or to exclude low priority traffic from taking certain links. Be sure only to add colors to links when necessary. If there are LSPs that should only use specific links in the network, or LSPs that should not use certain links, link coloring is an ideal solution. If requirements for link coloring don't exist, then adding it only constrains the network, adds additional configuration and planning overhead, and may even prevent LSPs from being created.

While the administrative group names and numerical values have only local significance, it is a best practice to use the same administrative group names, spellings, and numerical values on each router to ensure that the traffic-engineering database (TED) has a consistent look across all the routers in the MPLS domain. The TED is a common database containing information for each RSVP-enabled router (NodeID) in the domain. Information in the TED for each entry includes TE protocols running (ISIS/OSPF) and RSVP interface information, including interface destination (neighbor's lo0 address), static and reservable bandwidth, metric, and configured administrative groups (colors). Each router uses the TED to find the available paths through the network, including those paths with administrative constraints such as link-coloring. It is critical that all RSVP routers in the domain have an identical view of the TED. For example, the TED on glenlivet shows the link-coloring configured on dalwhinnie's ge-0/0/2.0 interface:

```
ps@glenlivet> show ted database extensive | find 10.200.86.5
NodeID: 10.200.86.5
  Type: Rtr, Age: 877 secs, LinkIn: 2, LinkOut: 2
  Protocol: OSPF(0.0.0.0)
    To: 10.200.86.6, Local: 192.168.86.6, Remote: 192.168.86.5
      Local interface index: 71, Remote interface index: 0
      Color: 0x2 gold
      Metric: 1
      Static BW: 1000Mbps
      Reservable BW: 1000Mbps
      Available BW [priority] bps:
         [0] 1000Mbps    [1] 1000Mbps    [2] 1000Mbps    [3] 1000Mbps
         [4] 1000Mbps    [5] 1000Mbps    [6] 1000Mbps    [7] 1000Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
```

```
    Encoding type: Packet
    Maximum LSP BW [priority] bps:
      [0] 1000Mbps    [1] 1000Mbps    [2] 1000Mbps    [3] 1000Mbps
      [4] 1000Mbps    [5] 1000Mbps    [6] 1000Mbps    [7] 1000Mbps
To: 10.200.86.7, Local: 192.168.86.30, Remote: 192.168.86.29
  Local interface index: 72, Remote interface index: 0
  Color: 0 <none>
  Metric: 1
  Static BW: 1000Mbps
  Reservable BW: 1000Mbps
  Available BW [priority] bps:
     [0] 1000Mbps    [1] 1000Mbps    [2] 1000Mbps    [3] 1000Mbps
     [4] 1000Mbps    [5] 1000Mbps    [6] 1000Mbps    [7] 1000Mbps
  Interface Switching Capability Descriptor(1):
    Switching type: Packet
    Encoding type: Packet
    Maximum LSP BW [priority] bps:
      [0] 1000Mbps    [1] 1000Mbps    [2] 1000Mbps    [3] 1000Mbps
      [4] 1000Mbps    [5] 1000Mbps    [6] 1000Mbps    [7] 1000Mbps
       . . .
       . . .
```

Glenlivet's TED shows information for the *NodeID: 10.200.86.5* (dalwhinnie's router-ID). It identifies dalwhinnie's ge-2/0/0.0 interface as *To: 10.200.86.6* (glenlivet's router-ID). It goes on to identify that interface as having the *Color: 0x2 gold* characteristic (0x2 is hex notation, equivalent to a decimal value of 2). Dalwhinnie's traffic engineering information appearing in glenlivet's TED illustrates the importance of keeping administrative group names and numerical values consistent. Imagine if glenlivet or any other router in this RSVP domain had different administrative group names assigned to different numerical values: it would be nearly impossible for a person to look at the TED and understand its data.

NOTE Although in our example the *gold* admin group actually has a value of 1, 1 is the second available value in an ordered list of 32 numbers with values 0-31.

## Strict and Loose LSP Hops

Earlier, in the *Failover* section of this chapter, the discussion of secondary LSPs started to examine the use of strict and loose hops defined in LSP paths: for a given MPLS path, a series of strict and/or loose hops can be defined for a primary and/or secondary LSP path, giving network engineers and planners a measure of control for primary and secondary paths.

When an RSVP LSP is configured for link-protection, the bypass LSP protecting a given RSVP link can also be defined with strict or loose hops. Shown below, the LSP *dalwhinnie-to-oban* is now configured for a primary path, defined with strict hops. Additionally, the RSVP bypass LSPs protecting dalwhinnie's RSVP interfaces are also now defined with strict hops, as you can see here:

```
ps@dalwhinnie> show configuration protocols mpls
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    link-protection;
    primary via-blair;
}
path via-blair {
    192.168.86.5 strict;
    192.168.86.9 strict;
    192.168.86.25 strict;
}
```

```
interface ge-0/0/2.0;
interface ge-0/0/3.0;

ps@dalwhinnie> show configuration protocols rsvp
interface ge-0/0/2.0 {
    link-protection {
        path {
            192.168.86.29 strict;
            192.168.86.1 strict;
            192.168.86.41 strict;
            192.168.86.46 strict;
        }
    }
}
interface ge-0/0/3.0 {
    link-protection {
        path {
            192.168.86.5 strict;
            192.168.86.45 strict;
            192.168.86.42 strict;
            192.168.86.2 strict;
        }
    }
}
```

Like before, the LSP's primary path is defined in the path stanza. Notice, however, how the next-hop bypass LSPs are defined: the next-hop bypass LSP protecting each RSVP interface is defined beneath that interface in the *protocols rsvp* stanza.

This configuration produces a specific path for both the primary path and bypass LSP for interface ge-0/0/2.0:

```
ps@dalwhinnie> show mpls lsp name dalwhinnie-to-oban detail ingress
Ingress LSP: 1 sessions

10.200.86.3
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-oban
  ActivePath: via-blair (primary)
  Link protection desired
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair       State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 3)
 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         10.200.86.6(flag=0x21) 192.168.86.5(flag=1 Label=300816) 10.200.86.1(flag=0x20)
192.168.86.9(Label=300816) 10.200.86.3(flag=0x20) 192.168.86.25(Label=3)
Total 1 displayed, Up 1, Down 0
ps@dalwhinnie>

ps@dalwhinnie> show mpls lsp bypass detail ingress
Ingress LSP: 2 sessions

10.200.86.6
  From: 10.200.86.5, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass->192.168.86.5
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300608
  Resv style: 1 SE, Label in: -, Label out: 300608
  Time left:    -, Since: Thu Jul 22 01:02:54 2010
```

```
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 1787 protocol 0
  Type: Bypass LSP
    Number of data route tunnel through: 1
    Number of RSVP session tunnel through: 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.29 (ge-0/0/3.0) 58 pkts
  RESV rcvfrom: 192.168.86.29 (ge-0/0/3.0) 58 pkts
  Explct route: 192.168.86.29 192.168.86.1 192.168.86.41 192.168.86.46
  Record route: <self> 192.168.86.29 192.168.86.1 192.168.86.41 192.168.86.46
Total 1 displayed, Up 1, Down 0
```

As expected, the RRO for each LSP matches the strict hops specified in the configuration.

With this capability, each router in the RSVP domain can specify its own path for the bypass LSPs that protect its RSVP interfaces. The glenlivet router has this configuration for the bypass protecting the glenlivet-blair link:

```
ps@glenlivet> show configuration protocols rsvp
interface ge-0/0/1.0 {
   link-protection {
      path {
         192.168.86.45 strict;
         192.168.86.21 strict;
         192.168.86.18 strict;
      }
   }
}
 . . .
 . . .
 . . .
```

```
ps@glenlivet> show mpls lsp bypass ingress detail
Ingress LSP: 1 sessions

10.200.86.1
  From: 10.200.86.6, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass->192.168.86.9
  Suggested label received: -, Suggested label sent: -
   . . .
   . . .
   . . .
  Explct route: 192.168.86.45 192.168.86.21 192.168.86.18
  Record route: <self> 192.168.86.45 192.168.86.21 192.168.86.18
Total 1 displayed, Up 1, Down 0
```

Here, the bypass LSP *Bypass->192.168.86.9* is intentionally configured to take a slightly longer route than optimal to demonstrate the control that network engineers have when configuring hops for primary paths or bypass LSPs. Figure 1.11 shows the primary path for the *dalwhinnie-to-oban* LSP and the configured strict-hop paths for the bypass LSPs protecting each link along the path. Configuring strict and/or loose hops for bypass LSPs give network engineers and capacity planners a large measure of control and predictability.

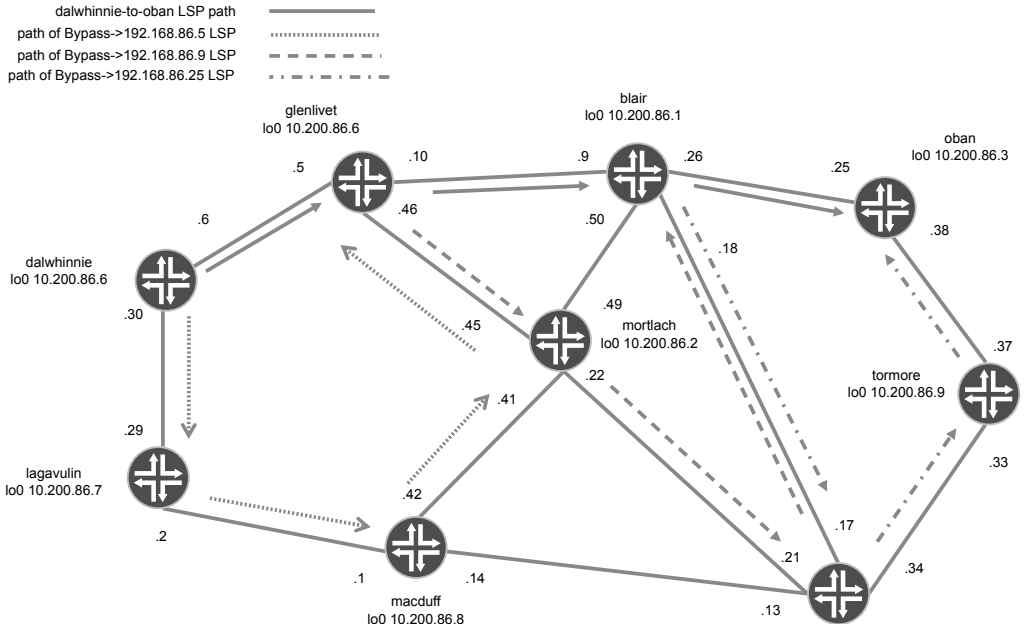



Figure 1.11 Strict-hop LSP Failover Traffic-engineering

## Route Table and LSP Integration

The traffic-engineering methods shown so far allow network engineers to control or influence the path that LSPs take through a network. Now we'll introduce the *inet.3* routing table and show how to control which traffic can take an LSP and how LSPs tie in with the *inet.0* and *inet.3* routing tables.

The inet.3 table contains the endpoints for a device's RSVP and LDP label switched paths. Its *sole* purpose is to give BGP another place to resolve its next-hops (in addition to inet.0). BGP, by default, then, is the only protocol that uses the inet.3 table and BGP traffic is the only type of traffic eligible to use LSPs. Later sections of this chapter examine how to change that default behavior, but that premise is the starting point.

## BGP Next-hop Installation

At this point, the dalwhinnie-to-oban LSP is established and running between those two routers (refer back to Figure 1.11). Dalwhinnie's inet.3 table contains the network's eight other lo0 addresses, each terminating an LDP or RSVP LSP, or both:

```
ps@dalwhinnie> show route table inet.3 | match metric
10.200.86.1/32     *[LDP/9] 00:03:40, metric 1
10.200.86.2/32     *[LDP/9] 00:03:40, metric 1
10.200.86.3/32     *[RSVP/7] 00:03:40, metric 3
                   [LDP/9] 00:03:40, metric 1
10.200.86.4/32     *[LDP/9] 00:03:40, metric 1
10.200.86.6/32     *[LDP/9] 00:03:39, metric 1
10.200.86.7/32     *[LDP/9] 00:03:39, metric 1
10.200.86.8/32     *[LDP/9] 00:03:40, metric 1
10.200.86.9/32     *[LDP/9] 00:03:40, metric 1
```

As expected, each of the other routers' lo0 appears in dalwhinnie's inet.3 table. Oban's lo0 (10.200.86.3) has an RSVP entry because of the dalwhinnie-to-oban RSVP LSP. Each address has an LDP entry because LDP is enabled on each core interface on each router, which spawns LDP LSPs.

The first example of BGP traffic using an LSP, shown in the output below, has oban (10.200.86.3) running iBGP with dalwhinnie and advertising 172.16.0/24 with an iBGP next-hop 10.200.86.3. Because 10.200.86.3 appears in dalwhinnie's inet.3 table, traffic from dalwhinnie bound for 172.16.0/24 takes the LSP to oban:

```
ps@dalwhinnie> show route 172.16.0/24

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.0.0/24      *[BGP/170] 2d 20:09:01, localpref 100, from 10.200.86.3
                    AS path: I
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
```

It is important to note that if the iBGP next-hop for 172.16.0/24 is an address on oban that is not in dalwhinnie's inet.3 table, then traffic for that route cannot take the RSVP LSP; the specific next-hop must appear in inet.3.

In the next case study (see Figure 1.12), the eBGP peer CE4 (192.168.90.14) is advertising 172.17.0/24 to oban. Oban is not rewriting its own IP address as the next-hop for that route before it advertises it via iBGP, so dalwhinnie sees the protocol next-hop for that route as 192.168.90.14.

```
ps@dalwhinnie> show route 172.17.0/24 detail

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
172.17.0.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Next hop type: Indirect
              Next-hop reference count: 3
              Source: 10.200.86.3
              Next hop type: Router, Next hop index: 554
              Next hop: 192.168.86.5 via ge-0/0/2.0, selected
              Protocol next hop: 192.168.90.14
 . . .
 . . .
 . . .
              Router ID: 10.200.86.3
```

Notice that the traffic destined for 172.17.0/24 does not take the *dalwhinnie-to-oban* LSP. This is because 192.168.90.14 (the BGP next-hop) is not present in dalwhinnie's inet.3 table.

In order to get dalwhinnie traffic to use the *dalwhinnie-to-oban* LSP en route to the destination 172.17.0/24, you can make use of the `install <route>` configuration in the *dalwhinnie-to-oban* LSP. Adding `install 192.168.90.12/30` to the *dalwhinnie-to-oban* LSP installs the 192.168.90.12/30 route to the inet.3 table, with the specific next hop as the *dalwhinnie-to-oban* LSP. This allows BGP to resolve the protocol next-hop (CE4 - 192.168.90.14) in the inet.3 table, which enables traffic bound for CE4's BGP routes to take the *dalwhinnie-to-oban* LSP when the LSP is available. Traceroute confirms that traffic bound for 172.17.0/24 takes that LSP after this configuration change:

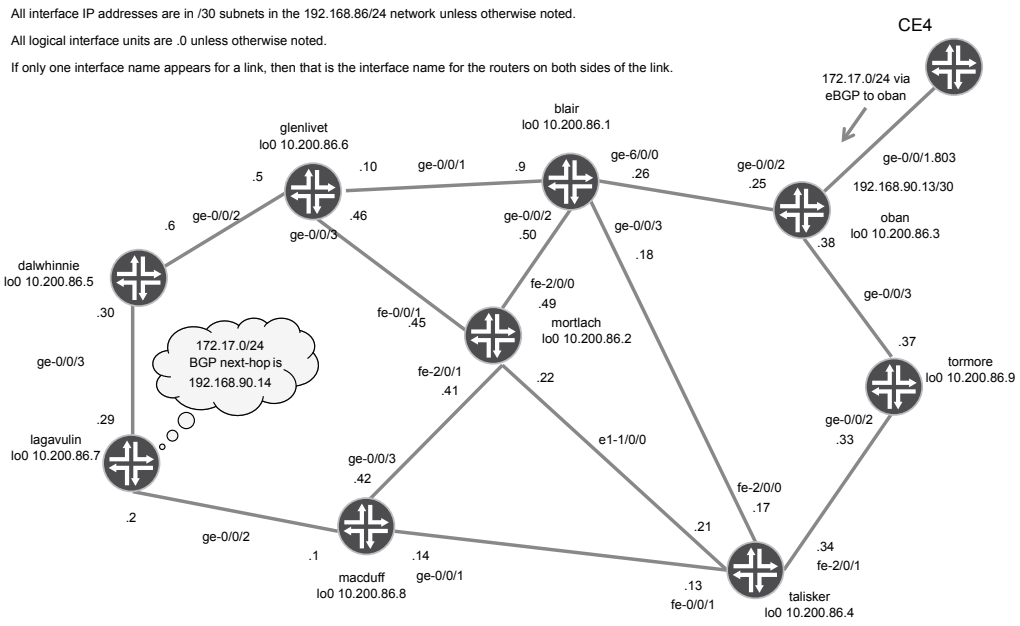



Figure 1.12 BGP Routes and LSPs

```
 [edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    install 192.168.90.12/30;
    link-protection;
    primary via-blair;
}
```

```
ps@dalwhinnie> show route 192.168.90.12/30

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.90.12/30   *[OSPF/10] 07:14:42, metric 4
                 > to 192.168.86.5 via ge-0/0/2.0

inet.3: 9 destinations, 10 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.90.12/30   *[RSVP/7] 00:00:43, metric 3
                 > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                   to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5

ps@dalwhinnie> show route 172.17.0/24 detail

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
172.17.0.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Next hop type: Indirect
              Next-hop reference count: 3
              Source: 10.200.86.3
              Next hop type: Router, Next hop index: 262142
```

```
            Next hop: 192.168.86.5 via ge-0/0/2.0 weight 0x1, selected
            Label-switched-path dalwhinnie-to-oban
            Label operation: Push 300928
            Next hop: 192.168.86.29 via ge-0/0/3.0 weight 0x8001
            Label-switched-path Bypass->192.168.86.5
            Label operation: Push 300928, Push 300656(top)
            Protocol next hop: 192.168.90.14
            . . .
            . . .
            . . .
            Router ID: 10.200.86.3


ps@dalwhinnie> traceroute 172.17.0.1 no-resolve
traceroute to 172.17.0.1 (172.17.0.1), 30 hops max, 40 byte packets
 1  192.168.86.5  4.675 ms  2.758 ms  2.843 ms
    MPLS Label=300928 CoS=0 TTL=1 S=1
 2  192.168.86.9  3.410 ms  4.945 ms  2.540 ms
    MPLS Label=300928 CoS=0 TTL=1 S=1
 3  192.168.86.25  5.750 ms  5.619 ms  7.720 ms
 4  192.168.90.14  4.575 ms !N  3.045 ms !N  2.865 ms !N

ps@dalwhinnie> show route table inet.3

inet.3: 9 destinations, 10 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.1/32     *[LDP/9] 1w0d 03:47:47, metric 1
                  > to 192.168.86.5 via ge-0/0/2.0, Push 299920
 . . .
 . . .
 . . .
                  > to 192.168.86.5 via ge-0/0/2.0, Push 300000
                    to 192.168.86.29 via ge-0/0/3.0, Push 300112
192.168.90.12/30   *[RSVP/7] 00:01:18, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
```

Notice that traffic destined for CE4's BGP routes can now take the *dalwhinnie-to-oban* LSP (our favorite LSP) en route to CE4. However, also note that traffic bound for 192.168.90.14 itself does *not* take the LSP (it takes the OSPF route in inet.0). This is because, by default, BGP can only use the inet.3 table to *resolve* next-hops; it cannot *forward* traffic to routes in inet.3 via LSP (remember, BGP can only use LSPs for a route whose *next-hop* is in inet.3, it can't send traffic via LSP to the inet.3 next-hops *themselves*).

In order to allow traffic bound for 192.168.90.12/30 to use an LSP, that RSVP /30 route must be moved from inet.3 to inet.0. Before we do that, let's get a baseline traceroute from dalwhinnie to CE4 (192.168.90.14):

```
ps@dalwhinnie> traceroute 192.168.90.14 no-resolve
traceroute to 192.168.90.14 (192.168.90.14), 30 hops max, 40 byte packets
 1  192.168.86.5  4.708 ms  4.491 ms  4.892 ms
 2  192.168.86.9  4.572 ms  3.854 ms  3.848 ms
 3  192.168.86.25  4.599 ms  4.878 ms  4.578 ms
 4  192.168.90.14  3.941 ms  3.737 ms  3.590 ms
```

The traceroute shows that traffic to that destination is not using an RSVP LSP.

Adding the `active` keyword to the `install 192.168.90.12/30` configuration within the LSP moves the RSVP route for 192.168.90.12/30 from the inet.3 table to the inet.0 table.

```
 [edit protocols mpls]
ps@dalwhinnie# show
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    install 192.168.90.12/30 active;
    link-protection;
    primary via-blair;
}
```

Traffic from dalwhinnie, bound for 192.168.90.12/30, now takes the label-switched-path *dalwhinnie-to-oban* when that LSP is available. Traceroute confirms that traffic takes that LSP en route to the destination once `active` is configured. Compare the previous `show route 192.168.90.12/30` output with the new output shown here, after the `active` keyword appears in the LSP configuration:

```
ps@dalwhinnie> show route 192.168.90.12/30

inet.0: 34 destinations, 35 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.90.12/30   *[RSVP/7] 00:00:26, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                  [OSPF/10] 1d 14:07:52, metric 4
                  > to 192.168.86.5 via ge-0/0/2.0

ps@dalwhinnie> traceroute 192.168.90.14 no-resolve
traceroute to 192.168.90.14 (192.168.90.14), 30 hops max, 40 byte packets
 1  192.168.86.5  4.492 ms  2.691 ms  2.833 ms
    MPLS Label=300944 CoS=0 TTL=1 S=1
 2  192.168.86.9  4.157 ms  4.080 ms  4.643 ms
    MPLS Label=300944 CoS=0 TTL=1 S=1
 3  192.168.86.25  4.677 ms  4.859 ms  4.624 ms
 4  192.168.90.14  10.693 ms  9.747 ms  3.859 ms
```

The configuration change moved the RSVP route 192.168.90.12 from inet.3 to inet.0, allowing traffic to that destination access to the RSVP LSP. Additionally, since BGP can use the inet.0 or inet.3 table for next-hop resolution, traffic bound for BGP routes with a protocol next-hop 192.168.0.14 still uses that LSP as well.

```
ps@dalwhinnie> show route 172.17.0.1/24

inet.0: 34 destinations, 35 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.17.0.0/24     *[BGP/170] 1d 07:34:55, localpref 100, from 10.200.86.3
                   AS path: 65432 I
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
```

Moving the 192.168.0.12/30 route from inet.3 to inet.0 allowed traffic bound for routes with that BGP next-hop to use the available LSP; it also allowed traffic bound for that destination itself to use the LSP.

The previous examples show how `install <route> (active)` can map specific routes to take a specific LSP on a local router. However, there are drawbacks to this method because of its similarity to a static route, its limited ability to scale, and the additional overhead it can add to the router's configuration on a per-route basis. Despite the necessary caution that should be used with this option, like a static route, it can also be a useful tool if applied properly.

At this point, let's remove the `install 192.168.90.12/30 active` configuration from the LSP in order to demonstrate the next concept:

```
 [edit protocols mpls]
ps@dalwhinnie# delete label-switched-path dalwhinnie-to-oban install
192.168.90.12/30
```

## Traffic-engineering bgp-igp

Earlier, during the introduction of the inet.3 table, this chapter showed that traffic from dalwhinnie, destined for BGP routes advertised with a protocol next-hop of 10.200.86.3 (oban), takes the available LSP to oban. However, traffic bound for oban's lo0 address itself cannot take the LSP because 10.200.86.3 only appears in the inet.3 table for BGP to use as a next-hop (by default, traffic to that address itself cannot take an available LSP). A show route and traceroute confirm this:

```
ps@dalwhinnie> show route 10.200.86.3

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32     *[OSPF/10] 1w3d 00:32:38, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0

inet.3: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32     *[RSVP/7] 01:24:24, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                  [LDP/9] 01:33:16, metric 1
                  > to 192.168.86.5 via ge-0/0/2.0, Push 300960

ps@dalwhinnie> traceroute 10.200.86.3 no-resolve
traceroute to 10.200.86.3 (10.200.86.3), 30 hops max, 40 byte packets
 1  192.168.86.5  675.632 ms  5.764 ms  4.069 ms
 2  192.168.86.9  5.025 ms  6.629 ms  4.293 ms
 3  10.200.86.3  3.808 ms  3.021 ms  3.996 ms
```

The resulting output shows that traffic to 10.200.86.3 does not take the LSP.

If the goal is to have all network internal traffic take an LSP path if one is available, there are several techniques within Junos to accomplish this. The first one we'll examine is setting `traffic-engineering bgp-igp` in the `[edit protocols mpls]` stanza. This configuration *empties* the contents of the local router's inet.3 table into the inet.0 table, allowing the IGP on dalwhinnie to see the LSP to oban. The inet.3 table is now empty, those routes having been moved to inet.0. Any traffic bound for any of the network's lo0 addresses takes an LSP, if available. Traffic from dalwhinnie to oban's lo0 now uses the available LSP, once again confirmed by the `show route` and `traceroute` output:

```
 [edit protocols mpls]
ps@dalwhinnie# show
traffic-engineering bgp-igp;
label-switched-path dalwhinnie-to-oban {
   to 10.200.86.3;
   link-protection;
   primary via-blair;
}
 . . .
 . . .
```

```
 . . .

[edit protocols mpls]
ps@dalwhinnie# run show route 10.200.86.3

inet.0: 34 destinations, 43 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32     *[RSVP/7] 00:00:28, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                  [LDP/9] 00:00:29, metric 1
                  > to 192.168.86.5 via ge-0/0/2.0, Push 300960
                  [OSPF/10] 00:00:29, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0

[edit protocols mpls]
ps@dalwhinnie# run traceroute 10.200.86.3 no-resolve
traceroute to 10.200.86.3 (10.200.86.3), 30 hops max, 40 byte packets
 1  192.168.86.5  3.955 ms  4.309 ms  2.823 ms
    MPLS Label=301040 CoS=0 TTL=1 S=1
 2  192.168.86.9  4.427 ms  2.806 ms  2.832 ms
    MPLS Label=300992 CoS=0 TTL=1 S=1
 3  10.200.86.3  8.533 ms  11.487 ms  12.444 ms

[edit protocols mpls]
ps@dalwhinnie# run show route table inet.3

[edit protocols mpls]
ps@dalwhinnie# run show route table inet.0 | match 10.200.*/32
10.200.86.1/32     *[LDP/9] 00:15:42, metric 1
10.200.86.2/32     *[LDP/9] 00:15:42, metric 1
10.200.86.3/32     *[RSVP/7] 00:15:41, metric 3
10.200.86.4/32     *[LDP/9] 00:15:42, metric 1
10.200.86.5/32     *[Direct/0] 4w0d 00:58:52
10.200.86.6/32     *[LDP/9] 00:15:41, metric 1
10.200.86.7/32     *[LDP/9] 00:15:41, metric 1
10.200.86.8/32     *[LDP/9] 00:15:42, metric 1
10.200.86.9/32     *[LDP/9] 00:15:42, metric 1
```

Notice that `traffic-engineering bgp-igp` is *not* configured on a per-LSP basis; this configuration change has a global effect. Also notice that the inet.3 table is truly empty, those routes now appearing in inet.0.

## Traffic-engineering bgp-igp-both-ribs

The drawback to using `traffic-engineering bgp-igp` is that in order to support VPNs, MPLS requires that the LSP egress address appear in inet.3, but since inet.3 is empty using the option above, another method must be used. The solution rests within another option of the traffic-engineering knob. Using the `bgp-igp-both-ribs` option *copies* the contents of inet.3 to inet.0 (instead of *emptying* inet.3). Configuring this option allows traffic to access the routes to LSP endpoints while still keeping those endpoints in inet.3 for VPN support:

```
 [edit protocols mpls]
ps@dalwhinnie# show
traffic-engineering bgp-igp-both-ribs;
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    link-protection;
    primary via-blair;
}
```

```
[edit]
ps@dalwhinnie# run show route table inet.3 | match /32
10.200.86.1/32    *[LDP/9] 00:00:20, metric 1
10.200.86.2/32    *[LDP/9] 00:00:20, metric 1
10.200.86.3/32    *[RSVP/7] 00:00:19, metric 3
10.200.86.4/32    *[LDP/9] 00:00:20, metric 1
10.200.86.6/32    *[LDP/9] 00:00:20, metric 1
10.200.86.7/32    *[LDP/9] 00:00:20, metric 1
10.200.86.8/32    *[LDP/9] 00:00:20, metric 1
10.200.86.9/32    *[LDP/9] 00:00:20, metric 1

[edit]
ps@dalwhinnie# run show route table inet.0 | match 10.200.*/32
10.200.86.1/32    *[LDP/9] 00:00:29, metric 1
10.200.86.2/32    *[LDP/9] 00:00:29, metric 1
10.200.86.3/32    *[RSVP/7] 00:00:28, metric 3
10.200.86.4/32    *[LDP/9] 00:00:29, metric 1
10.200.86.5/32    *[Direct/0] 4w0d 01:05:53
10.200.86.6/32    *[LDP/9] 00:00:29, metric 1
10.200.86.7/32    *[LDP/9] 00:00:29, metric 1
10.200.86.8/32    *[LDP/9] 00:00:29, metric 1
10.200.86.9/32    *[LDP/9] 00:00:29, metric 1
```

Inet.3's routes now appear in both the inet.3 and inet.0 tables.

## Traffic-engineering mpls-forwarding

Using `traffic-engineering bgp-igp-both-ribs` causes the inet.3 routes to supplant the IGP routes to those destinations in the inet.0 table, due to the inet.3 routes' better (lower) preference. Configure `traffic-engineering mpls-forwarding` to keep IGP routes active in the inet.0 table for routing purposes. This option may be necessary for any legacy policies that require the IGP routes to be available. Using this option, the LSP endpoints still stay active for forwarding purposes. The following output shows the *before* and *after* effects of this configuration:

```
 [edit]
ps@dalwhinnie# run show route 10.200.86.3

inet.0: 34 destinations, 43 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32    *[RSVP/7] 00:07:51, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                  [LDP/9] 00:07:52, metric 1
                  > to 192.168.86.5 via ge-0/0/2.0, Push 300960
                  [OSPF/10] 00:30:06, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0

inet.3: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32    *[RSVP/7] 00:07:51, metric 3
                  > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                    to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                  [LDP/9] 00:07:52, metric 1
                  > to 192.168.86.5 via ge-0/0/2.0, Push 300960

[edit]
ps@dalwhinnie# delete protocols mpls traffic-engineering bgp-igp-both-ribs
```

```
[edit]
ps@dalwhinnie# set protocols mpls traffic-engineering mpls-forwarding

[edit]
ps@dalwhinnie# commit
commit complete

[edit]
ps@dalwhinnie# run show route 10.200.86.3

inet.0: 34 destinations, 43 routes (33 active, 0 holddown, 1 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32    @[OSPF/10] 00:31:57, metric 3
                   > to 192.168.86.5 via ge-0/0/2.0
                  #[RSVP/7] 00:00:10, metric 3
                   > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                     to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                   [LDP/9] 00:00:11, metric 1
                   > to 192.168.86.5 via ge-0/0/2.0, Push 300960

inet.3: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32    *[RSVP/7] 00:00:10, metric 3
                   > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                     to 192.168.86.29 via ge-0/0/3.0, label-switched-path Bypass->192.168.86.5
                   [LDP/9] 00:00:11, metric 1
                   > to 192.168.86.5 via ge-0/0/2.0, Push 300960
```

For the purposes of further demonstration in subsequent sections of this chapter, dalwhinnie will be rolled back to the `traffic-engineering bgp-igp-both-ribs` configuration under the [edit protocols mpls] stanza.

```
[edit]
ps@dalwhinnie# show protocols mpls
traffic-engineering bgp-igp-both-ribs;
. . .
. . .
```

## IGP Shortcuts

At this point, all traffic from dalwhinnie bound for any of the network's lo0 addresses takes an LDP or RSVP LSP if available. One last step is necessary, though, to meet the requirement that all network internal traffic take an available LSP. Traffic to IGP interfaces still does not use available LSPs as a next-hop. For instance, the route from dalwhinnie to oban's ge-0/0/3.0 interface (192.168.86.38) still uses the IGP route. A traceroute confirms:

```
[edit]
ps@dalwhinnie# run show route 192.168.86.38

inet.0: 34 destinations, 43 routes (33 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.86.36/30   *[OSPF/10] 00:02:00, metric 4
                    > to 192.168.86.5 via ge-0/0/2.0
```

```
[edit]
ps@dalwhinnie# run traceroute 192.168.86.38 no-resolve
traceroute to 192.168.86.38 (192.168.86.38), 30 hops max, 40 byte packets
 1  192.168.86.5  4.082 ms  4.697 ms  4.052 ms
 2  192.168.86.9  4.798 ms  6.081 ms  4.835 ms
 3  192.168.86.38  5.058 ms  3.122 ms  3.678 ms
```

This behavior may present some operational difficulties as it may appear to some operators that an LSP to oban does not exist or is not operational. To change this next-hop behavior, configure `shortcuts` in the `[edit protocols ospf traffic-engineering]` stanza or configure `set protocols isis traffic-engineering family inet shortcuts` if your network runs ISIS. This book's lab network uses OSPF. The following output shows the effects of the addition of `shortcuts`: all of dalwhinnie's network internal traffic can now use LSP paths if they are available:

```
 [edit protocols mpls]
ps@dalwhinnie# show
traffic-engineering bgp-igp-both-ribs;
label-switched-path dalwhinnie-to-oban {
    to 10.200.86.3;
    link-protection;
    primary via-blair;
}

[edit]
ps@dalwhinnie# show protocols ospf
traffic-engineering {
    shortcuts;
}

[edit]
ps@dalwhinnie# commit
commit complete
```

```
[edit]
ps@dalwhinnie# run traceroute 192.168.86.38 no-resolve
traceroute to 192.168.86.38 (192.168.86.38), 30 hops max, 40 byte packets
 1  192.168.86.5  3.190 ms  2.776 ms  3.606 ms
    MPLS Label=301168 CoS=0 TTL=1 S=1
 2  192.168.86.9  2.676 ms  4.863 ms  2.578 ms
    MPLS Label=301072 CoS=0 TTL=1 S=1
 3  192.168.86.38  3.719 ms  3.799 ms  3.820 ms
```

NOTE Unlike OSPF, ISIS supports `traffic-engineering` and creation of a traffic-engineering database by default. So adding the `shortcuts` configuration requires explicit configuration of `traffic-engineering` in ISIS.

NOTE Our example already had `protocols mpls traffic-engineering` configured before turning on the IGP shortcuts. If `protocols mpls traffic-engineering` is not configured, the shortcuts configuration moves all IGP addresses that could use an LSP as a next-hop into the inet.3 table, where they can be used as BGP next-hops. In our example, because dalwhinnie already had `traffic-engineering bgp-igp-both-ribs` configured, all IGP routes that could use an LSP as a next-hop were populated into the inet.0 table instead.

The addition of `traffic-engineering shortcuts` allows traffic to the internal IGP interface address 192.168.86.38 to take an available LSP from dalwhinnie to that destination.

Use of the `traffic-engineering` configuration knobs under the MPLS and IGP protocols has a wide effect over which traffic uses the available LSPs on the router, allowing these methods to scale very well. The tradeoff is that they offer less control over which prefix/prefix-range uses a particular LSP: it's a classic shotgun (`traffic-engineering` knobs) versus sniper (`install` and `active` in the LSP configuration) approach. The appropriate approach for a given network depends on the network's size, potential for growth, operational considerations, and network requirements.

The methods for route table integration mentioned thus far also limit their behavior to the router where those configurations reside. For instance, dalwhinnie has `traffic-engineering` configured under the MPLS and OSPF protocols so that it can use LSPs for network internal and transit traffic. That configuration, however, does not affect routing decisions made by other routers in the network, so let's discuss options that may affect routing decisions made by IGP neighbors.

## Advertising LSPs Directly Into the IGP

Another option available to you to allow internal network traffic to take available LSPs is advertising label-switched-paths directly into the IGP with a metric. In the following example, lagavulin has LSPs to oban and tormore advertised directly into OSPF, just like an interface. When advertising LSPs into IGP, it is a best practice to have equally metriced LSPs in IGP in the reverse direction as well. In the example here, oban and tormore should have LSPs to lagavulin advertised into their OSPF configurations as well (not shown):

```
 [edit protocols ospf]
ps@lagavulin# show
traffic-engineering;
area 0.0.0.0 {
    interface ge-0/0/2.0 {
        interface-type p2p;
    }
    interface ge-0/0/3.0 {
        interface-type p2p;
    }
    interface lo0.0;
    label-switched-path lagavulin-to-oban {
        metric 2;
    }
    label-switched-path lagavulin-to-tormore {
        metric 2;
    }
}
```

```
ps@lagavulin> show ospf neighbor
Address         Interface           State     ID            Pri  Dead
192.168.86.1    ge-0/0/2.0          Full      10.200.86.8     128    32
192.168.86.30   ge-0/0/3.0          Full      10.200.86.5     128    35
10.200.86.3     lagavulin-to-oban     Full      10.200.86.3       0     0
10.200.86.9     lagavulin-to-tormore  Full      10.200.86.9       0     0

ps@lagavulin> show ospf interface
Interface         State   Area          DR ID         BDR ID        Nbrs
ge-0/0/2.0        PtToPt  0.0.0.0       0.0.0.0       0.0.0.0          1
ge-0/0/3.0        PtToPt  0.0.0.0       0.0.0.0       0.0.0.0          1
lagavulin-to-oban   PtToPt  0.0.0.0       0.0.0.0       0.0.0.0          1
lagavulin-to-tormorePtToPt  0.0.0.0       0.0.0.0       0.0.0.0          1
lo0.0             DR      0.0.0.0       10.200.86.7    0.0.0.0          0

ps@lagavulin>
```

A full mesh of RSVP LSPs among routers, with each LSP advertised into IGP, changes the inet.0 table considerably, by allowing the IGP to access and advertise LSPs. Output from `show route` on lagavulin confirms that traffic to lo0 and OSPF interface addresses takes LSPs:

```
ps@lagavulin> show route 10.200.86.3

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3/32     *[OSPF/10] 00:12:56, metric 2
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban

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

10.200.86.3/32     *[RSVP/7] 00:12:51, metric 2
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban

ps@lagavulin> show route 192.168.86.33

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.86.32/30   *[OSPF/10] 00:09:43, metric 3
                  > to 192.168.86.1 via ge-0/0/2.0
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore
```

Traffic from lagavulin to oban's lo0 interface and tormore's ge-0/0/2.0 interface takes available LSPs.

Unlike the traffic-engineering configuration used previously in this chapter, this method may also alter the routing decisions made by IGP neighbors: because the IGP advertises the LSPs just like it would any other IGP interface, the router advertises the LSPs in its OSPF Type 1 LSAs or ISIS TLVs. IGP neighbors may start to use the local router's LSPs to move network internal traffic. Advertising LSPs to the IGP is useful for larger networks because it scales well. Drawbacks include IGP peers changing their routing decisions based on the existence of the LSP as an available path; not to mention that great understanding of network behavior is necessary to effectively implement this option.

TIP It is considered a best practice to NOT implement both IGP shortcuts and LSP advertisement into IGP.

## Policy-based LSP-mapping

The final method of traffic-engineering that this chapter examines is *policy-based LSP-mapping*. To demonstrate this, the RSVP LSPs have been removed from lagavulin's OSPF stanza, leaving the two logical network interfaces actively running OSPF. Lagavulin now has two RSVP LSPs to oban, each with a different primary path through the network, as shown configured here:

```
 [edit protocols ospf]
ps@lagavulin# show
traffic-engineering;
area 0.0.0.0 {
   interface ge-0/0/2.0 {
      interface-type p2p;
```

```
            }
            interface ge-0/0/3.0 {
                interface-type p2p;
            }
            interface lo0.0 {
                passive;
            }
        }

        [edit protocols mpls]
        ps@lagavulin# show
        label-switched-path lagavulin-to-tormore {
            to 10.200.86.9;
        }
        label-switched-path lagavulin-to-oban {
            to 10.200.86.3;
            primary via-blair;
        }
        label-switched-path lagavulin-to-oban-prime {
            to 10.200.86.3;
            primary via-talisker;
        }
        path via-blair {
            10.200.86.1 loose;
        }
        path via-talisker {
            10.200.86.4 loose;
        }
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
```

Just recently, oban started advertising the four /24 routes in the 172.16.0/22 range via iBGP to lagavulin. The goal in this next method of traffic-engineering is to force the 172.16.0/24 and 172.16.1/24 networks to use the *lagavulin-to-oban* LSP and to force the 172.16.2/24 and 172.16.3/24 networks to use the *lagavulin-to-oban-prime* LSP. The output here shows each route's path selection before implementing the policy-based LSP-mapping.

```
ps@lagavulin> show route protocol bgp range 172.16.0.0/22

inet.0: 36 destinations, 36 routes (36 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.0.0/24      *[BGP/170] 00:00:25, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime
172.16.1.0/24      *[BGP/170] 00:00:25, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime
172.16.2.0/24      *[BGP/170] 00:00:25, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime
172.16.3.0/24      *[BGP/170] 00:00:25, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime
```

At this point, all routes prefer the *lagavulin-to-oban-prime* LSP. Figure 1.13 shows each LSP's path.

NOTE The results above suggest that all traffic to routes within 172.16.0/22 will take the lagavulin-to-oban-prime LSP. However, by default, when there are multiple equal cost paths to the same destination for an active route, the Junos OS uses a hash algorithm to choose one of the next-hops to install into the forwarding-table

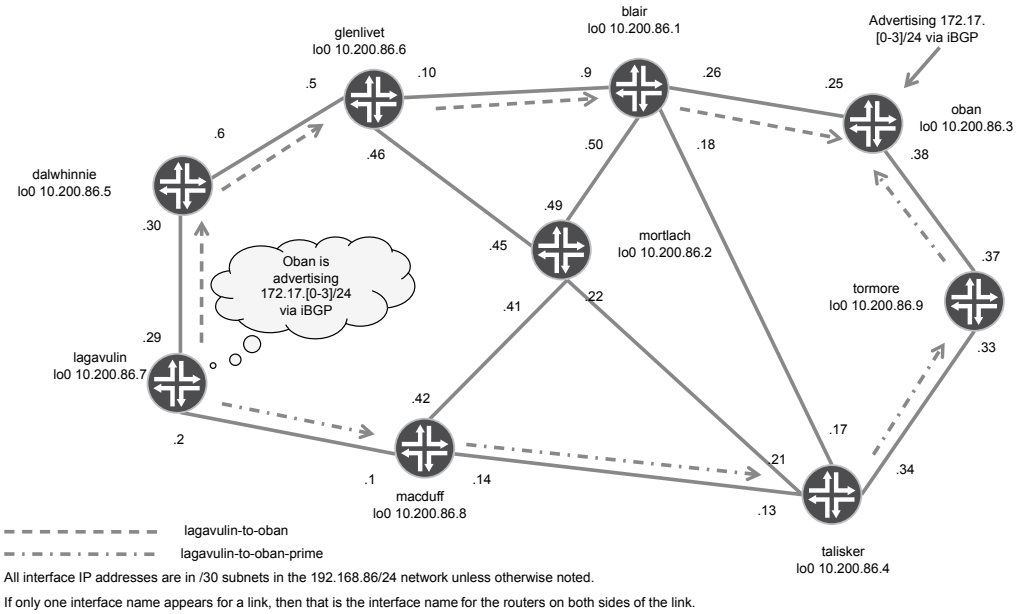



Figure 1.13 Policy-mapping Is Often Used on Multiple LSPs Between the Same Routers

The first step in policy-based LSP-mapping involves crafting a policy to map specific routes, or a range of routes, to specific LSPs. The policy shown here accomplishes this for the four advertised prefixes of interest:

```
 [edit policy-options]
ps@lagavulin# show
policy-statement map-routes-to-oban-lsps {
   term 10 {
      from {
         protocol bgp;
         neighbor 10.200.86.3;
         route-filter 172.16.0.0/24 orlonger;
         route-filter 172.16.1.0/24 orlonger;
      }
      then {
         install-nexthop lsp lagavulin-to-oban;
         accept;
      }
   }
   term 20 {
      from {
         protocol bgp;
         neighbor 10.200.86.3;
         route-filter 172.16.2.0/24 orlonger;
         route-filter 172.16.3.0/24 orlonger;
      }
      then {
         install-nexthop lsp lagavulin-to-oban-prime;
         accept;
      }
   }
}
```

Once complete, apply the policy in the `[edit routing-options forwarding-table]` stanza.

```
 [edit routing-options]
ps@lagavulin# show
 . . .
 . . .
 . . .
forwarding-table {
   export map-routes-to-oban-lsps;
}
```

The policy is applied as an *export* policy because the policy is applied from the point of view of the route-table to the forwarding-table.

The following output shows the desired prefixes mapped to take the appropriate LSP. This is a rather simple example, but can be scaled up to accommodate many routes and BGP peers if appropriate:

```
ps@lagavulin> show route protocol bgp range 172.16.0.0/22

inet.0: 36 destinations, 36 routes (36 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.0.0/24      *[BGP/170] 3d 18:53:04, localpref 100, from 10.200.86.3
                    AS path: I
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
172.16.1.0/24      *[BGP/170] 01:06:32, localpref 100, from 10.200.86.3
                    AS path: I
                  > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban
172.16.2.0/24      *[BGP/170] 01:06:32, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime
172.16.3.0/24      *[BGP/170] 01:06:32, localpref 100, from 10.200.86.3
                    AS path: I
                    to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-oban-prime

ps@lagavulin>
```

As expected, the 172.16.0/24 and 172.16.1/24 routes now explicitly prefer the *lagavulin-to-oban* LSP; the other two routes explicitly prefer the *lagavulin-to-oban-prime* LSP.

Policy-based LSP mapping can be effective for managing traffic between two endpoints where multiple equal-cost LSPs exist. Just as with the `install <route> (active)` method, this method may not scale well: managing the unique policies on each router may start to become burdensome each time the network grows or carries more traffic. However, when used properly, policy-based LSP mapping can give network engineers very granular control over traffic flows.

## TE Summary

This section covered the multiple TE options available on a Juniper router. Which option(s) are best to implement depends on the specific requirements, growth potential, and traffic patterns on a specific network. Table 1.1 summarizes key points about each method for your convenience.

Table 1.1 Summary of TE Options

| TE Method | Description | Affects | Advantages | Drawbacks |
|---|---|---|---|---|
| LSP metric | Assigns a static metric for an LSP, regardless of underlying interface IGP metrics. | Fixed metric for LSP, regardless of path. | Provides a stable, predictable cost. | May result in suboptimal path selection. |
| Link coloring | Requires or forbids LSPs from taking specific links; applied network-wide. | LSP path | Ability to reserve links for high-priority traffic or spread traffic evenly through the network links. | Can overcomplicate the network and restrict LSP formation if implemented improperly. |
| Strict/loose hops | Controls/influences LSP primary or secondary path on a per-LSP basis. | LSP path | Ability to control LSP path. | May limit failover options or prevent LSP from establishing. |
| Install route (active) | Maps specific prefixes/ prefix ranges to take specific LSPs on a per-LSP basis; applied on a per-router basis. | Traffic eligible to take LSP; local routing decisions. | Ability to control which LSP a prefix uses; allows non-BGP prefixes to use LSP paths. | Similar behavior to static route; care must be taken not to overload a particular LSP; may not scale well as network and traffic grow. |
| Traffic-engineering bgp-igp | Empties inet.3 into inet.0, allowing inet.3 entries to be used for forwarding; applied on a per-router basis. | Traffic eligible to take LSP; local routing decisions. | Allows non-BGP traffic to take LSPs to LSP endpoints; scales well. | Does not support MPLS services* (covered in Chapter 2). |
| Traffic-engineering bgp-igp-both-ribs | Copies inet.3 into inet.0, allowing inet.3 entries to be used for forwarding; supports MPLS services*; applied on a per-router basis. | Traffic eligible to take LSP; local routing decisions. | Allows non-BGP traffic to take LSPs to LSP endpoints; scales well. | Begin to lose ability to control which prefixes should be mapped to a specific RSVP LSP. |
| Traffic-engineering mpls-forwarding | Copies inet.3 into inet.0, but still retains IGP routing entries for policy processing; supports MPLS services*; applied on a per-router basis. | Traffic eligible to take LSP; local routing decisions.. | Allows non-BGP traffic to take LSPs to LSP endpoints; allows support for legacy policies that may require that IGP routes be available for processing; scales well. | Begin to lose ability to control which prefixes should be mapped to a specific RSVP LSP. |
| IGP shortcuts | Allows IGP routes to take available LSPs; applied on a per-router basis. | Traffic eligible to take LSP; local routing decisions. | Allows all internal traffic to take LSPs; scales well. | Diminished ability to determine which prefixes should be mapped to a specific RSVP LSP. |
| LSPs in IGP | IGP uses/advertises LSPs just as it does an IGP interface; applied on a per-router basis. Can affect routing decisions by other IGP peer routers. | Traffic eligible to take LSP; network routing decisions. | Allows network traffic to use LSPs, even traffic bound for external destinations; scales well. | Network-wide impact requires careful planning and implementation. |

| | | | | |
|---|---|---|---|---|
| Policy-based LSP-mapping | Maps specific prefixes/prefix ranges to take specific LSPs via export policy applied to forwarding-table on a per-router basis. | Traffic eligible to take LSP; local routing decisions. | Useful for managing load-balancing when multiple equal-cost LSPs exist between two routers; depending on implementation, may be more scalable than install route configuration as a single policy can be updated versus each LSP with install route. | Great care must be taken when creating policy to ensure that policy does not overload a particular LSP; may not scale well as network and traffic grow. |

## LSP Bandwidth Management

If you are unable to augment your network with both the hardware and physical connectivity necessary to ensure that any given link won't become saturated, or the network has become so large that it becomes operationally difficult to find and utilize available capacity, it is imperative to correctly manage LSP bandwidth in order to help maximize your existing assets. LSP bandwidth management is a method of traffic engineering: it spreads the traffic over the available resources. While bandwidth management is a subset of traffic engineering, it is covered in this dedicated section because there are many features and techniques available. Because bandwidth management focuses on optimizing existing resources instead of adding capacity along the most optimal path, some traffic may not take the most optimal path through the network. However, in a resource-constrained environment it may be more important to ensure that traffic reaches its destination rather than arriving there via the most optimal path.

### Static LSP Bandwidth

It is possible to allocate a specific amount of bandwidth for an LSP so that when the ingress router signals the LSP, a specific amount of bandwidth on each link becomes reserved for that LSP. In the example output below, the LSPs *lagavulin-to-oban* and *lagavulin-to-oban-prime* each reserve 150Mbps of bandwidth. This is a constraint when the LSP is signaled; if a path does not exist with the available bandwidth, then the LSP fails to come up.

```
[edit protocols mpls]
ps@lagavulin# show
label-switched-path lagavulin-to-oban {
    to 10.200.86.3;
    bandwidth 150m;
    primary via-blair;
}
label-switched-path lagavulin-to-oban-prime {
    to 10.200.86.3;
    bandwidth 150m;
    primary via-talisker;
}
 . . .
 . . .
```

It can be a fairly useful, if static, tool: it reserves bandwidth on each transit link, but manual updates to each LSP's bandwidth are still necessary if the traffic on the LSP

continues to grow. Additionally, it may be inefficient to statically reserve a large amount of bandwidth for an LSP for future growth if that LSP only needs to carry a small amount of traffic in the near- to mid-term because that bandwidth reservation may consume bandwidth needed for other LSPs. This option may be acceptable for an LSP that carries a constant, easily predictable amount of traffic that is not expected to grow very much, if at all. Happily, though, the Junos OS offers more dynamic options to manage LSP bandwidth.

## Automatic Bandwidth Allocation

Automatic bandwidth allocation, unlike static LSP bandwidth, allows an RSVP LSP to dynamically adjust its bandwidth reservation based on the amount of traffic it is carrying. Traffic is moved from the old path to the new path (with the modified bandwidth allocation) in a make-before-break fashion: the ingress router signals a new LSP path with the modified bandwidth and if the path successfully comes up, the router moves the traffic from the old to new path. If there is not adequate bandwidth for the new path, then the current path and its current bandwidth reservation remain. This means that automatic bandwidth allocation does not impact traffic when a new path is signaled. For instance, an LSP can be configured for minimal reserved initial bandwidth and then use automatic allocation to adjust its reserved bandwidth based on actual usage without experiencing packet loss during a transition.

NOTE The *allocated* bandwidth is not necessarily the maximum amount of traffic that the LSP can carry. Rather, the allocated bandwidth is simply the amount of *reserved* bandwidth on any given link that the LSP traverses. If the links along the LSP's path are not saturated with actual traffic or reserved bandwidth, then the LSP can carry as much traffic as necessary until there is bandwidth contention on any given link. In order to prioritize certain traffic during periods of congestion, class-of-service must be implemented, which is beyond the scope of this book.

The first step in enabling automatic bandwidth allocation is to configure MPLS statistics with the *auto-bandwidth* option. The *interval*, specified in seconds, is the amount of time used to calculate the average bandwidth usage.

```
[edit protocols mpls]
ps@lagavulin# show
statistics {
   file auto-bw;
   interval 600;
   auto-bandwidth;
}
```

Here, the interval is set for 600 seconds (10 minutes). This configuration allows lagavulin to collect the traffic statistics needed for each individual LSP's periodic adjustments.

Once the router is configured to collect MPLS statistics, automatic bandwidth allocation must be enabled on each desired LSP. To enable automatic bandwidth allocation on an individual LSP, configure the *auto-bandwidth* keyword.

```
[edit protocols mpls]
ps@lagavulin# show
statistics {
   file auto-bw;
   interval 600;  ## interval to calculate average bandwidth usage; default is 300 seconds
   auto-bandwidth;
```

```
}
label-switched-path lagavulin-to-oban {
    to 10.200.86.3;
    auto-bandwidth {
        adjust-interval 7200;  ## bandwidth reallocation interval; default is 86,400 seconds
        adjust-threshold 20;   ## specifies the sensitivity of the adjust-interval (optional)
        minimum-bandwidth 64k; ## optional – min bandwidth reservation for LSP
        maximum-bandwidth 150m;  ## optional – hard max limit bandwidth reservation for LSP
    }
    primary via-blair;
}
```

TIP It is considered a best practice to explicitly specify the values of the MPLS statistics interval and the LSP's auto-bandwidth adjust-interval, even if you are using the default values. Doing so helps to act as a reminder or a notice to Operations and Engineering as to what type of behavior to expect from automatic bandwidth allocation and when to expect it.

The configuration above shows automatic bandwidth allocation configured for the LSP *lagavulin-to-oban*. Enabling this feature for the LSP only requires the *auto-bandwidth* keyword, but other options in the *auto-bandwidth* hierarchy are configured to conform to best practices and to demonstrate the different capabilities of the available options. As configured above, here is the behavior of the automatic bandwidth allocation for the *lagavulin-to-oban* LSP:

- The average bandwidth on each LSP is measured every 600 seconds.
- The configured *adjust-interval* value of 7200 means that the *lagavulin-to-oban* LSP can adjust its bandwidth allocation every 7200 seconds based on the traffic statistics for that period.
- The LSP's *adjust-threshold* is 20%. If the current bandwidth allocation for this LSP is 100Mbps and the bandwidth demand increases to 110 Mbps or decreases to 90 Mbps (10%), the LSP's bandwidth allocation is not adjusted. But if the bandwidth demand increases to 125 Mbps or decreases to 75 Mbps (25%), then the LSP bandwidth is adjusted to 125 Mbps or 75 Mbps, respectively.

The LSP will have a minimum of 64kbps allocated bandwidth and a hard maximum of 150 Mbps reserved bandwidth.

TIP When configuring the LSP's auto-bandwidth adjust-interval and the MPLS statistics interval, it is a best practice to set the LSP's auto-bandwidth adjust-interval at no less than three times the value of the MPLS statistics interval. Or, to put it mathematically, 3*(MPLS statistics interval) < = (LSP auto-bandwidth adjust-interval). This minimizes unnecessary LSP resignaling by ensuring that the LSP has a minimum of three MPLS statistics batches to draw upon before deciding if it needs to adjust its bandwidth allocation.

WARNING As of this writing, some of the Junos documentation up to and including 10.0 incorrectly indicates that the LSP auto-bandwidth adjust-interval should be no MORE than three times the value of the MPLS statistics interval (3*(MPLS statistics interval) > = (LSP auto-bandwidth adjust-interval)). This is being remedied, but the remedy date for corrected documentation is unknown at the time of this writing.

## Monitor-only

In the event that you only want to monitor an LSP's bandwidth but don't want to have the router make automatic allocation adjustments, use the *monitor-bandwidth* keyword in the LSP's auto-bandwidth stanza:

```
 [edit protocols mpls]
ps@lagavulin# show label-switched-path lagavulin-to-oban-prime
to 10.200.86.3;
bandwidth 150m;
auto-bandwidth {
    adjust-interval 86400;
    monitor-bandwidth;
}
primary via-talisker;
```

In the above output, lagavulin's *lagavulin-to-oban-prime* LSP is monitoring its allocated bandwidth, but is not modifying its allocated bandwidth. This is called *passive bandwidth utilization monitoring*.

## Manual Bandwidth Adjustment Trigger

Junos allows a user to manually trigger an automatic bandwidth allocation adjustment for all LSPs configured for auto-bandwidth or just specific LSPs with that configuration. This can be useful when testing automatic bandwidth allocation in a lab or when the LSP is configured for passive bandwidth utilization monitoring and you want to initiate a bandwidth adjustment. Initiate an immediate automatic bandwidth allocation adjustment with the operational command *request mpls lsp adjust-autobandwidth*.

```
ps@lagavulin> request mpls lsp adjust-autobandwidth ?
Possible completions:
  <[Enter]>          Execute this command
  name               Regular expression for LSP names to match
  |                 Pipe through a command

ps@lagavulin> request mpls lsp adjust-autobandwidth
```

Without the optional LSP name, the command invokes the adjustment in all LSPs configured for auto-bandwidth. If the user includes the optional regular expression for the LSP name, then all LSPs configured for auto-bandwidth whose names contain the specified regular expression automatically adjust their bandwidth if the criteria for a bandwidth adjustment have been met for the individual LSP.

NOTE When a manual trigger for automatic bandwidth adjustment is used, the adjust-interval timer on each affected LSP is not reset.

## Auto-bandwidth in Action – An Example

This next section shows an example of auto-bandwidth in action. Consider the following configuration on lagavulin:

```
 [edit protocols mpls]
ps@lagavulin# show
statistics {
    file auto-bw;
    interval 600;
```

```
        auto-bandwidth;
    }
    label-switched-path lagavulin-to-oban {
        to 10.200.86.3;
        auto-bandwidth {
            adjust-interval 7200;
            adjust-threshold 20;
            minimum-bandwidth 64k;
            maximum-bandwidth 150m;
                        }
        primary via-blair;
    }
```

Initially, 100Mbps of traffic is sent over the *lagavulin-to-oban* LSP (the bandwidth shown for the LSP is slightly lower than this due to the averaging over time).

```
ps@lagavulin> show mpls lsp name lagavulin-to-oban detail
Ingress LSP: 6 sessions

10.200.86.3
  From: 10.200.86.7, State: Up, ActiveRoute: 0, LSPname: lagavulin-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
  Max AvgBW util: 90.4322Mbps, Bandwidth Adjustment in 6423 second(s).
  Overflow limit: 0, Overflow sample count: 0
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair       State: Up
    Priorities: 7 0
    Bandwidth: 90.4322Mbps
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
 192.168.86.30 S 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.30 192.168.86.5 192.168.86.9 192.168.86.25
 . . .
 . . .
```

The amount of traffic is then increased to 150Mbps. Notice the *Max AvgBW util* value increasing above the value of the LSP's reserved bandwidth.

```
ps@lagavulin> show mpls lsp name lagavulin-to-oban extensive
Ingress LSP: 6 sessions

10.200.86.3
  From: 10.200.86.7, State: Up, ActiveRoute: 0, LSPname: lagavulin-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
  Max AvgBW util: 110.722Mbps, Bandwidth Adjustment in 6317 second(s).
  Overflow limit: 0, Overflow sample count: 0
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair       State: Up
    Priorities: 7 0
    Bandwidth: 90.4322Mbps
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
 192.168.86.30 S 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
```

```
        192.168.86.30 192.168.86.5 192.168.86.9 192.168.86.25
 . . .
 . . .
```

A manual request for automatic bandwidth allocation adjustment is issued (this has the same effect as the *adjust-interval* timer expiring):

```
ps@lagavulin> request mpls lsp adjust-autobandwidth name "lagavulin-to-oban"

ps@lagavulin> show mpls lsp name lagavulin-to-oban extensive
Ingress LSP: 6 sessions

10.200.86.3
  From: 10.200.86.7, State: Up, ActiveRoute: 0, LSPname: lagavulin-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
  Max AvgBW util: 111.402Mbps, Bandwidth Adjustment in 6302 second(s).
  Overflow limit: 0, Overflow sample count: 0
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-blair      State: Up
    Priorities: 7 0
    Bandwidth: 110.722Mbps
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 4)
 192.168.86.30 S 192.168.86.5 S 192.168.86.9 S 192.168.86.25 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         192.168.86.30 192.168.86.5 192.168.86.9 192.168.86.25
   11 Jan  5 08:05:25.662 Record Route:  192.168.86.30 192.168.86.5 192.168.86.9 192.168.86.25
   10 Jan  5 08:05:25.662 Up
    9 Jan  5 08:05:25.662 Automatic Autobw adjustment succeeded
    8 Jan  5 08:05:25.629 Originate make-before-break call
 . . .
 . . .

ps@lagavulin> show log messages | match "bandwidth changed"
Jan  5 08:05:25  lagavulin rpd[1075]: RPD_MPLS_PATH_BANDWIDTH_CHANGE: MPLS path via-blair (lsp
lagavulin-to-oban) bandwidth changed, path bandwidth 110721928 bps
Jan  5 08:05:26  lagavulin rpd[1075]: RPD_MPLS_LSP_BANDWIDTH_CHANGE: MPLS LSP lagavulin-to-oban
bandwidth changed, lsp bandwidth 110721928 bps
Jan  6 05:41:26  lagavulin mgd[97769]: UI_CMDLINE_READ_LINE: User 'ps', command 'show log messages |
match "bandwidth changed" '
```

Notice that the allocated bandwidth for the LSP changed from 90.4Mbps to 110.7Mbps. The LSP adjusted its bandwidth because the *Max AvgBW* value was greater than 20% of the previous average when the manual request was triggered (keep in mind that the manual *adjust-autobandwidth* request evokes the same behavior as the *adjust-interval* timer expiring). The LSP initiated a make-before-break call to signal the new path before moving the traffic. Understand that these values represent the maximum average traffic, not the maximum traffic, over the measurement periods; since this data comes from a lab environment and uses the *adjust-autobandwidth* manual adjustment trigger, the time periods for traffic measurement are artificially shorter than those for a production network.

WARNING Automatic bandwidth allocation adjustment does not *create* bandwidth, but can only act as a tool to help *optimize* existing bandwidth and assist in understanding when it is time to augment the network. It is certainly not a substitute for proper network planning, but rather a tool to assist in efficient utilization of existing resources.

## Planning for Sudden LSP Bandwidth Increases

Typically an LSP's *adjust-interval* is configured for longer periods of time, sometimes hours or days (in this example it's configured for 7200 seconds, or two hours). During that time, there may be a large increase in traffic on the LSP, potentially causing congestion and or packet loss; this is typically not an acceptable condition that should go on for hours. The *adjust-threshold-overflow-limit* feature is designed to deal with this type of situation. The *lagavulin-to-oban* LSP shows this additional configuration below:

```
[edit protocols mpls]
ps@lagavulin# show
statistics {
    file auto-bw;
    interval 600;
    auto-bandwidth;
}
label-switched-path lagavulin-to-oban {
    to 10.200.86.3;
    auto-bandwidth {
        adjust-interval 7200;
        adjust-threshold 20;
        minimum-bandwidth 64k;
        maximum-bandwidth 150m;
        adjust-threshold-overflow-limit 2;
    }
    primary via-blair;
}
```

This LSP's *adjust-threshold-overflow-limit* is set for two MPLS statistics intervals. When the *adjust-threshold-overflow-limit* is configured, after each MPLS statistics interval (600 seconds in our example) the following conditions are checked for the LSP:

- Did the average bandwidth utilization for the LSP during the interval exceed the current maximum average bandwidth utilization?
- Has the change in the maximum average bandwidth utilization exceeded the configured adjust-threshold?

These conditions can often be difficult to understand. Let's consider an example. An LSP's maximum average bandwidth shows up in the *show mpls lsp name <lsp-name> detail* output once *auto-bandwidth* is configured (see output below). In the example below, the current *Max AvgBW util* value is 90.4Mbps. At the end of every MPLS statistics interval, the LSP's reserved bandwidth (25.7Mbps in the example below) is compared to the *Max AvgBW util* value for that period; if the *Max AvgBW util* is greater than the LSP bandwidth, point (1) is true. If the *change* in *Max AvgBW util* exceeds the adjust threshold (20% in our example), then point (2) is also true.

```
ps@lagavulin> show mpls lsp name lagavulin-to-oban detail
Ingress LSP: 6 sessions

10.200.86.3
  From: 10.200.86.7, State: Up, ActiveRoute: 0, LSPname: lagavulin-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Random
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
```

```
 Max AvgBW util: 90.4322Mbps, Bandwidth Adjustment in 6423 second(s).
 Overflow limit: 2, Overflow sample count: 0
 Encoding type: Packet, Switching type: Packet, GPID: IPv4
*Primary   via-blair      State: Up
   Priorities: 7 0
   Bandwidth: 25.6762Mbps
```

So if conditions (1) and (2) above are true, then the sample is considered to be one bandwidth overflow sample. The *adjust-threshold-overflow-limit* statement allows you to configure a limit on the number of *consecutive* overflow samples: when the limit is reached, the LSP's current adjust-interval timer is expired and the LSP adjusts its bandwidth. Once the adjustment occurs, the adjust-interval timer then starts again from zero.

It is important to understand the difference between the *adjust-interval* and *adjust-threshold-overflow-limit*. The adjust-interval is best suited to handle more gradual increases or decreases in bandwidth over a longer period of time. The adjust-threshold-overflow-limit, on the other hand, is best suited to deal with more dramatic increases in bandwidth requirements over a relatively shorter period of time.

NOTE As of the time of this writing, the *adjust-threshold-overflow-limit* can only increase the LSP's bandwidth allocation; the ability to decrease LSP bandwidth allocation based on sudden decreases in bandwidth is not available. Downward LSP resizing, if necessary, is done at the end of each *adjust-interval* period.

## Most-fill/least-fill/random

When multiple equal-cost paths exist to the same destination, it is possible to configure an explicit CSPF (constrain shortest path first – the algorithm used to determine an LSP's path) tie-breaker based on the ratio of a path's available bandwidth on a link to total reservable bandwidth on the link. A method for tie-breaking is necessary because an LSP's RRO (the sequenced list of routers that the LSP must transit) can only have one path. The tie-breaking options are *most-fill*, *least-fill*, and *random*. Potential LSP paths with differing amounts of available and reservable bandwidth are compared by a ratio of each link's available bandwidth divided by its reservable bandwidth. The available/reservable ratio for a link is known as the *available bandwidth ratio*.

```
available bandwidth ratio = (available bandwidth)/(reservable bandwidth)
```

The output from *show rsvp interface* displays an interface's available and reservable bandwidth.

```
 [edit protocols mpls]
ps@dalwhinnie# run show rsvp interface
RSVP interface: 2 active
               Active Subscr- Static     Available   Reserved    Highwater
Interface   State resv   iption  BW         BW         BW         mark
ge-0/0/2.0  Up       1   100%  1000Mbps    850Mbps     150Mbps     150Mbps
                    ge-0/0/3.0  Up        0   100%  1000Mbps    1000Mbps    0bps       0bps
```

In the example output above for ge-0/0/2.0 on dalwhinnie, ge-0/0/2.0 is showing a reservation of 150Mbps, 850Mbps of available bandwidth, and 1000Mbps of RSVP reservable bandwidth (shown as *static BW* in the output). So interface ge-0/0/2.0 has an available bandwidth ratio of 850Mbps/1000Mbps or 0.85.

It's possible to cap the amount of bandwidth that the RSVP protocol can reserve on

an interface by specifying a bandwidth value in the [protocols rsvp] stanza. Here is shown dalwhinnie.ge-0/0/2.0 capped at 500Mbps for RSVP bandwidth reservation.

```
[edit protocols rsvp]
ps@dalwhinnie# show interface ge-0/0/2.0
bandwidth 500m;
link-protection;


[edit protocols]
ps@dalwhinnie# run show rsvp interface ge-0/0/2.0
                Active Subscr- Static     Available   Reserved    Highwater
Interface   State resv   iption  BW         BW         BW         mark
ge-0/0/2.0  Up       1   100%  500Mbps     350Mbps     150Mbps     150Mbps
```

Notice how the static bandwidth changed from 1000Mbps (default value for *static BW* is the interface's physical bandwidth) to 500Mbps; this modification changes ge-0/0/2.0's available bandwidth ratio to 350Mbps/500Mbps (0.70) in this example.

Table 1.2 lists each CSPF tie-breaker method and the general network behavior associated with each.

Table 1.2 Most-fill/Least-fill/Random Behavior

| Option | General Behavior |
|---|---|
| Most-fill | Prefer the LSP path with the link that has the highest utilization (which means the least amount of available bandwidth, so the lowest available bandwidth ratio). This generally results in links with some traffic becoming close to full before less-utilized links are consumed.<br>Note: The lowest available bandwidth ratio of a given path is the single lowest available bandwidth ratio belonging to any one of the links in the path. This is an important point to understand – the determination of the path with the most-fill is determined by the single link in the path with the lowest bandwidth ratio. |
| Least-fill | Prefer the LSP path with the least utilized links (larger amount of available bandwidth, so a higher available bandwidth ratio). This generally results in a more even distribution of traffic across the network. |
| Random | Randomly select one of the equal-cost paths. As long as one of the equal-cost paths has enough reservable bandwidth, it is a candidate for the additional traffic. This is the default setting. |

Configure the CSPF equal-cost tie breaker at the [edit protocols mpls <lsp-name>] hierarchy level. For example, lagavulin's *lagavulin-to-oban* and *lagavulin-to-oban-prime* LSPs are configured for most-fill: the router uses the path containing the link with the lowest available bandwidth ratio until that interface is filled up.

```
 [edit protocols mpls]
ps@lagavulin# show
 . . .
 . . .
label-switched-path lagavulin-to-oban {
   to 10.200.86.3;
   most-fill;
   auto-bandwidth {
      adjust-interval 7200;
```

```
            adjust-threshold 20;
            minimum-bandwidth 64k;
            maximum-bandwidth 150m;
            adjust-threshold-overflow-limit 2;
        }
        primary via-blair;
    }
    label-switched-path lagavulin-to-oban-prime {
        to 10.200.86.3;
        bandwidth 150m;
        most-fill;
        auto-bandwidth {
            adjust-interval 86400;
        }
        primary via-talisker;
    }
     . . .
     . . .
     . . .
```

TIP When using most-fill/least-fill/random for the CSPF tie-breaker, it is generally a best practice to use only one method across all the LSPs in the network.

The CSPF tie-breaker method for an LSP can be confirmed by checking the detailed output of the `show mpls lsp <lsp-name> detail` operational command:

```
ps@lagavulin> show mpls lsp name lagavulin-to-oban detail
Ingress LSP: 6 sessions

10.200.86.3
  From: 10.200.86.7, State: Up, ActiveRoute: 1, LSPname: lagavulin-to-oban
  ActivePath: via-blair (primary)
  LoadBalance: Most-fill
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
```

TIP Configuring many options and features common to many or all of the LSPs may become tedious. Junos *groups* (a type of template) can simplify configuration by applying common configurations to selected LSPs. The example below applies *auto-bandwidth* attributes to all LSPs.

```
 [edit]
ps@lagavulin# show
## Last changed: 2011-02-15 02:41:30 UTC
version 10.0R3.10;
groups {
    LSP-ATTRIBUTES {
        protocols {
            mpls {
                label-switched-path <*> {
                    auto-bandwidth {
                        adjust-interval 7200;
                adjust-threshold 20;
                minimum-bandwidth 64k;
                        maximum-bandwidth 150m;
                        }
                    }
                }
            }
```

```
    }
}
apply-groups LSP-ATTRIBUTES;
 . . .
 . . .


[edit]
ps@lagavulin# show protocols mpls
statistics {
    file auto-bw;
    interval 600;
    auto-bandwidth;
}
 . . .
 . . .
label-switched-path lagavulin-to-dalwhinnie {
    to 10.200.86.5;
}
 . . .
 . . .


[edit]
ps@lagavulin# show protocols mpls | display inheritance
statistics {
    file auto-bw;
    interval 600;
    auto-bandwidth;
}
 . . .
 . . .
label-switched-path lagavulin-to-dalwhinnie {
    to 10.200.86.5;
    ##
    ## 'auto-bandwidth' was inherited from group 'LSP-ATTRIBUTES'
    ##
    auto-bandwidth {
        ##
        ## '7200' was inherited from group 'LSP-ATTRIBUTES'
        ##
        adjust-interval 7200;
        ##
        ## '20' was inherited from group 'LSP-ATTRIBUTES'
        ##
        adjust-threshold 20;
        ##
        ## '64k' was inherited from group 'LSP-ATTRIBUTES'
        ##
        minimum-bandwidth 64k;
        ##
        ## '150m' was inherited from group 'LSP-ATTRIBUTES'
        ##
        maximum-bandwidth 150m;
    }
}
 . . .
 . . .


[edit]
ps@lagavulin# run show mpls lsp name lagavulin-to-dalwhinnie detail
Ingress LSP: 5 sessions
```

```
10.200.86.5
  From: 10.200.86.7, State: Up, ActiveRoute: 1, LSPname: lagavulin-to-dalwhinnie
  ActivePath:  (primary)
  LoadBalance: Random
  Autobandwidth
  MinBW: 64kbps MaxBW: 150Mbps
  AdjustTimer: 7200 secs AdjustThreshold: 20%
  Max AvgBW util: 112bps, Bandwidth Adjustment in 216 second(s).
  Overflow limit: 0, Overflow sample count: 0
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                State: Up
    Priorities: 7 0
    Bandwidth: 64kbps
    SmartOptimizeTimer: 180
```

When looking at the configuration for the LSP, the configurations from the applied group do not show up unless the | *display inheritance* is added to the CLI *show* command. The *show mpls lsp* output confirms that the *lagavulin-to-dalwhinnie* LSP has the attributes configured in the applied group.

MORE? An in-depth discussion on Junos *groups* is beyond the scope of this book; more information on this Junos feature can be found in the *Day One* book *Configuring Junos Basics*. A free download is available at www.juniper.net/dayone and eBooks are available at Amazon and on Apple's iBookstore.

## Summary

This chapter covered concepts and different possible characteristics of an MPLS core network in a brief fashion. It is not meant to be an all-inclusive reference for each available concept in the core. But, having said that, you should now have a basic understanding of the differences between LDP and RSVP, what an LSP is, the choices and tradeoffs for each method of RSVP LSP failover, and the many different methods of traffic-engineering and LSP bandwidth management, as well as a few base configurations for each concept.

Now that you have an understanding of the MPLS concepts in the network core, let's move on to Chapter 2 and its concepts for MPLS services, the next logical building block.

# Chapter 2

## MPLS Services Concepts

MPLS services leverage the capabilities of an MPLS core to provide network users with additional resources. The three main MPLS services categories are Layer 3 VPN (L3VPN), Layer 2 VPN (L2VPN), and VPLS. This chapter covers the capabilities and basic mechanics of each of these three services.

## Introduction

In general, there are three types of routers in an MPLS services scenario: *customer edge*, *provider edge*, and *provider core*. The customer edge (CE) router offers direct end-user connectivity as it aggregates the traffic for specific users or for users at a specific site. The CE connects to the provider edge (PE) router whose role it is to aggregate traffic from multiple CEs, apply any appropriate class-of-service (CoS) treatment, provide the logical separation of data between the users, and then send the data to the provider core (P) routers.

As shown in Figure 2.1, PE routers typically have ingress and egress LSPs but not transit LSPs (accommodating transit LSPs is typically a role of the P router). The P routers are the MPLS backbone routers and their role is to provide an MPLS backbone capable of moving data from one PE to the destination PE(s), which usually means that P routers primarily act as conduits for LSPs to transit en route to a PE. The destination PE places data into its appropriate logical segment within the router, applies appropriate CoS treatment, and passes the data on to the CE.

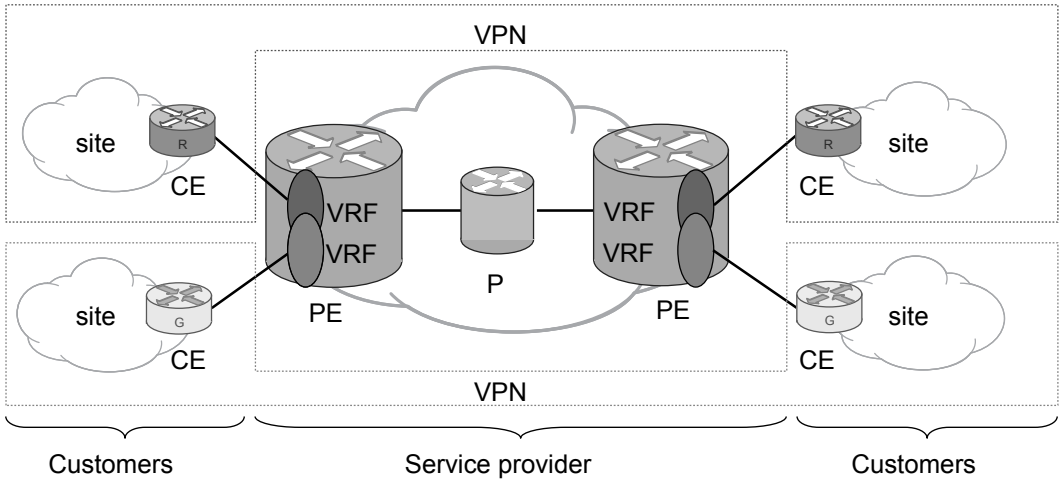



Figure 2.1 VPN Router Roles

Figure 2.2 shows a generic network layout for CE, PE, and P routers, that we'll use in this chapter as the generic starting point for a discussion on MPLS services. Depending on network size, user requirements, potential for growth, and a host of other factors, some router roles are often combined. A very common example of this is for

a single router to have both a PE and P role. Network architecture decisions such as this are examined in Chapter 3. Note that due to this network's size and requirements, Figure 2.2's PE routers also fulfill some roles of P routers in that they act as paths for transit LSPs.

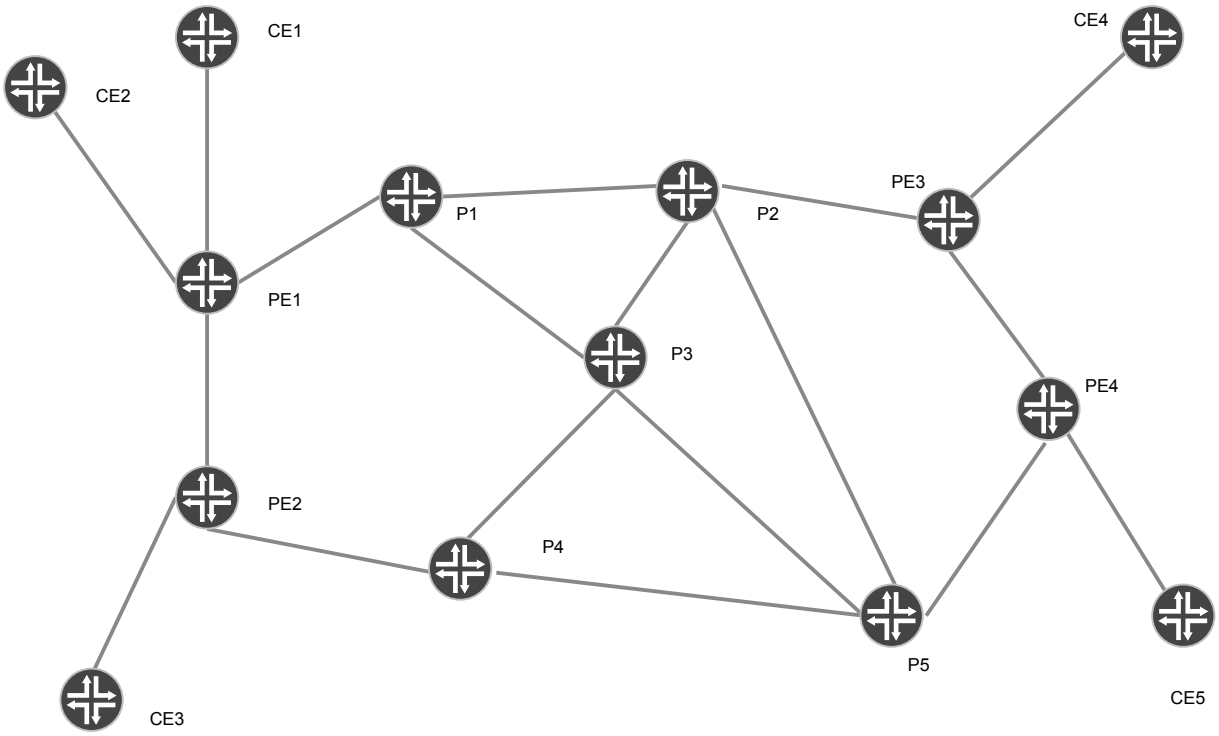



Figure 2.2 Generic MPLS Scenario Layout

## Layer 3 VPN

Layer 3 VPN (L3VPN) offers a way for groups of users to communicate via Layer 3 over an MPLS core without the communication being exposed to general Internet traffic or other VPN traffic. A Layer 3 VPN is made up of a group of sites that share common routing information and whose connectivity between the sites is controlled by a group of policies. The basic premise of an L3VPN is a *virtual route and forwarding* (VRF) instance. In plain terms, this is a logical segment in the PE router where the CE-facing interface(s), virtual routing tables, and policies that control connectivity reside. Additionally, any protocols running between the PE and CE are also configured in the VRF routing-instance (from here on it will simply be referred to as a *VRF*). Figure 2.3 shows a visual representation of a PE router with three VRFs, each its own routing table. Each VRF's routing table contains only routes for that VRF; in Junos each VRF's routing-table is named `<routing-instance name>.inet.0`.

MORE? The L3VPN is based on the IETF RFC 4364. For more information, see http://tools.ietf.org/html/rfc4364.

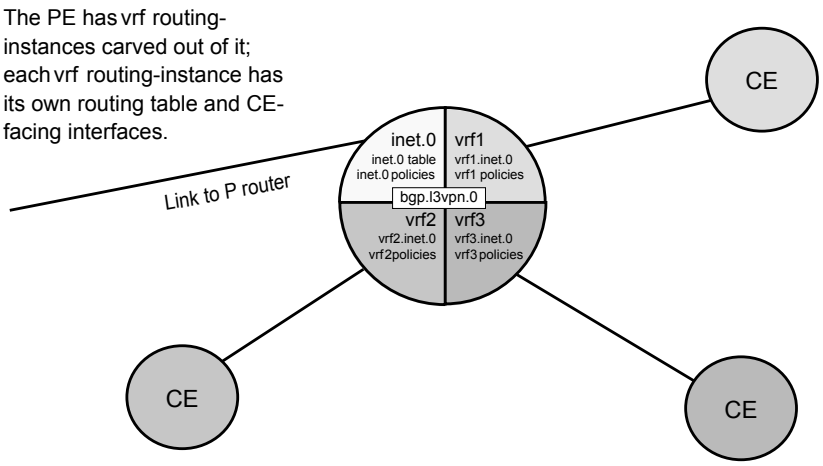



Figure 2.3 Logical VRF Representation

## L3VPN Control Plane

Common VRFs on different PEs exchange routes with each other via multiprotocol BGP (MP-BGP). Specifically, they use the *family inet-vpn* configuration to advertise and receive the NLRI. The inet-vpn configuration can be configured at the `[edit protocols bgp]`, `[edit protocols bgp group <group-name>]`, or `[edit protocols bgp group <group-name> neighbor <neighbor-id>]` hierarchy depending on which groups and neighbors need to advertise and receive the L3VPN NLRI:

```
ps@dalwhinnie> show configuration protocols bgp
family inet {
    any;
}
family inet-vpn {
    any;
}
```

In general terms, the PE where the VRF instance resides advertises that VRF's routing table to its MP-BGP peers. Each VRF's prefixes are kept unique by the VRF's unique *route-distinguisher* (RD); different VRFs are required to have RD values. The RD is prepended to the VRF's routes, thereby keeping each prefix unique when it is advertised to iBGP neighbors and re-advertised by route reflectors. Prepending the RD allows different user groups in different VRFs to use overlapping address schemes, ensuring each prefix remains unique due to the unique RD prepended to each.

A PE receiving L3VPN routes receives them via the *bgp.l3vpn.0 routing-table*. This table stores all VPN-IPv4 unicast routes received from other PE routers. If and when the route is accepted into the local router's VRF, the RD is removed and the route is placed into the local VRF routing-table.

It's best to use an example to assist in understanding the role of the RD, so examine Figure 2.4, which illustrates a sample L3VPN layout.

In Figure 2.4, the CE1 and CE5 routers are in the *spruce* L3VPN and the CE2, CE3, and CE4 routers are in the *aspen* L3VPN. Dalwhinnie receives four routes in the bgp.l3vpn.inet.0 table: two instances of 172.17.0.0/24, 192.168.90.12/30, and 192.168.90.16/30. Notice that both CE4 and CE5 are advertising 172.17.0/24 to

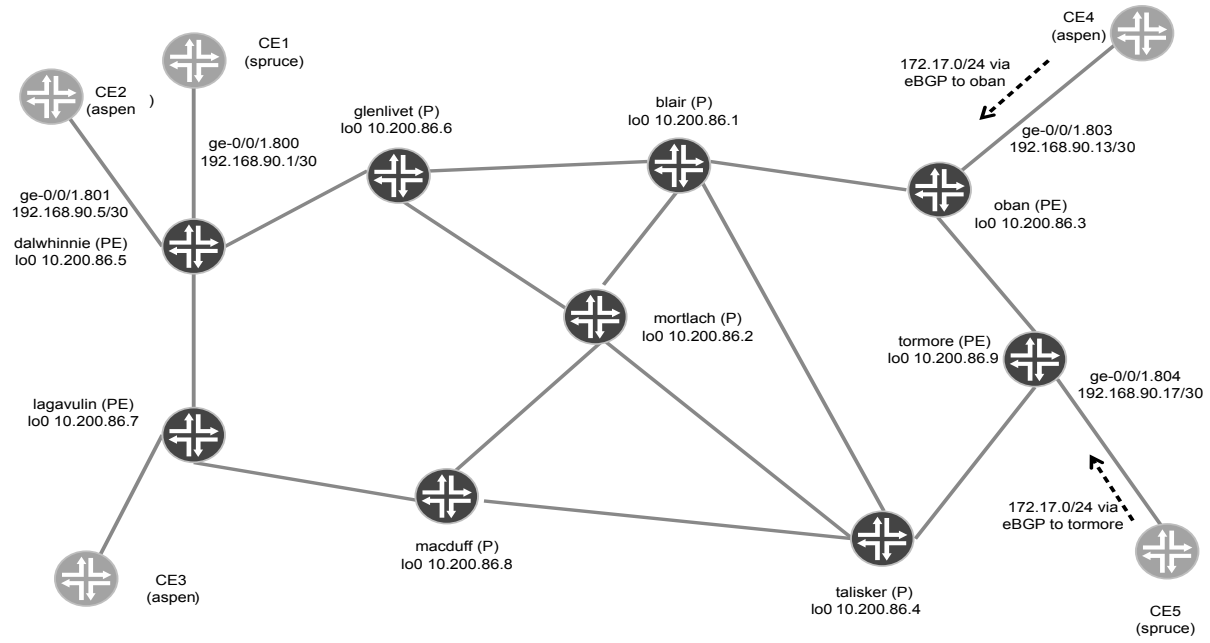



Figure 2.4 MPLS Network with L3VPN Service

their respective PE routers. The output here shows dalwhinnie's bgp.l3vpn.0 routing-table:

```
ps@dalwhinnie> show route table bgp.l3vpn.0

bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.90.14:20:172.17.0.0/24
                *[BGP/170] 2d 13:53:39, localpref 100, from 10.200.86.3
                  AS path: 65432 I
                 > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                  to 192.168.86.29 via ge-0/0/3.0, label-switched-path dalwhinnie-to-oban
192.168.90.14:20:192.168.90.12/30
                *[BGP/170] 2d 13:53:39, localpref 100, from 10.200.86.3
                  AS path: I
                 > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-oban
                  to 192.168.86.29 via ge-0/0/3.0, label-switched-path dalwhinnie-to-oban
192.168.90.18:30:172.17.0.0/24
                *[BGP/170] 00:02:01, localpref 100, from 10.200.86.9
                  AS path: 65433 I
                 > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-tormore
192.168.90.18:30:192.168.90.16/30
                *[BGP/170] 2d 13:54:27, localpref 100, from 10.200.86.9
                  AS path: I
                 > to 192.168.86.5 via ge-0/0/2.0, label-switched-path dalwhinnie-to-tormore
```

Notice how the two 172.17.0.0/24 routes are kept unique in this table because each has a different route-distinguisher: *192.168.90.14:20* or *192.168.90.18:30*. This effectively makes each 172.17.0.0/24 route unique when placed into the bgp.l3vpn.0 table.

At this point, dalwhinnie's bgp.l3vpn.0 table shows four VPN-IPv4 routes it is

receiving from other PEs. How do dalwhinnie's two configured L3VPN routing-instances know which routes to import? The routing-instances use the route-target (RT) to make this determination.

The configurations shown here are very basic configurations for L3VPN routing-instances:

```
ps@dalwhinnie> show configuration routing-instances
aspen {
    instance-type vrf;
    interface ge-0/0/1.801;
    route-distinguisher 192.168.90.6:20;
    vrf-target target:100:200;
    vrf-table-label;
    protocols {
        bgp {
            group ce2 {
                neighbor 192.168.90.6 {
                    peer-as 65432;
                }
            }
        }
    }
}
spruce {
    instance-type vrf;
    interface ge-0/0/1.800;
    route-distinguisher 192.168.90.2:30;
    vrf-target target:100:300;
    vrf-table-label;
    protocols {
        bgp {
            group ce1 {
                neighbor 192.168.90.2 {
                    peer-as 65433;
                }
            }
        }
    }
}
```

The *route target* is a BGP extended community with the format *target:x:y*. The policies that control communication between a common VRF's sites typically allow or deny prefixes based on the route-target attached to the route. In this configuration, routing-instance *aspen* is looking for routes that have a route-target value `target:100:200` while *spruce* is looking for `target:100:300`. The `show route detail` output for the 172.17.0/24 route shown here is very telling:

```
ps@dalwhinnie> show route 172.17.0.0/24 detail

aspen.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
172.17.0.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Route Distinguisher: 192.168.90.14:20
              . . .
              . . .
              . . .
              Communities: target:100:200
              Import Accepted
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.3
              Primary Routing Table bgp.l3vpn.0
```

```
spruce.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

172.17.0.0/24 (1 entry, 1 announced)
      *BGP    Preference: 170/-101
              Route Distinguisher: 192.168.90.18:30
              . . .
              . . .
              . . .
              Communities: target:100:300
              Import Accepted
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.9
              Primary Routing Table bgp.l3vpn.0

bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

192.168.90.14:20:172.17.0.0/24 (1 entry, 0 announced)
      *BGP    Preference: 170/-101
              Route Distinguisher: 192.168.90.14:20
              . . .
              . . .
              . . .
              Communities: target:100:200
              Import Accepted
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.3
              Secondary Tables: aspen.inet.0

192.168.90.18:30:172.17.0.0/24 (1 entry, 0 announced)
      *BGP    Preference: 170/-101
              Route Distinguisher: 192.168.90.18:30
              . . .
              . . .
              . . .
              Communities: target:100:300
              Import Accepted
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.9
              Secondary Tables: spruce.inet.0
```

This output confirms the following:

- The aspen VRF accepts the 172.17.0/24 route with the route-target target:100:200 and installs it into aspen.inet.0.
- The spruce VRF accepts the 172.17.0/24 route with the route-target target:100:300 and installs it into spruce.inet.0.
- The route-distinguisher (RD) keeps routes unique in the bgp.l3vpn.0 table, even if there are routes with overlapping space.
- The route-target (RT) determines which routes are accepted from bgp.l3vpn.0 and placed into a local VRF's routing table.

NOTE Understand that VRF routing-instance names are only locally significant. A VRF's membership in an L3VPN is dependent on its associated RTs and import/export policies, not its name. However, it is a best practice to have a common instance name for a given L3VPN across all routers.

You should take any time necessary to understand these points because there is often confusion about the role of the RT and RD. Understanding the difference between the two (the last two items in the list above) is crucial to a complete understanding of L3VPN.

## L3VPN Forwarding Plane

With a basic understanding of the roles of MP-BGP, the RT, and RD, and how they affect L3VPN route distribution, let's examine how L3VPN traffic travels from the source CE to the ingress PE, through the core, and through the egress PE en route to the destination CE.

The CE2 router is attempting to send a packet to CE4's 192.168.90.14 address, which it receives via the eBGP session with dalwhinnie (reference previous Figure 2.4). This `show route detail` output provides a wealth of information as to how the L3VPN forwarding plane works:

```
ps@dalwhinnie> show route advertising-protocol bgp 192.168.90.6

aspen.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
* 192.168.90.12/30       Self                              I


ps@dalwhinnie> show route 192.168.90.14 table aspen detail

aspen.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
192.168.90.12/30 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Route Distinguisher: 192.168.90.14:20
              Next hop type: Indirect
              Next-hop reference count: 6
              Source: 10.200.86.3
              Next hop type: Router, Next hop index: 262147
              Next hop: 192.168.86.5 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path dalwhinnie-to-oban
              Label operation: Push 16, Push 301328(top)
              Next hop: 192.168.86.29 via ge-0/0/3.0 weight 0x8001
              Label-switched-path dalwhinnie-to-oban
              Label operation: Push 16, Push 301328, Push 301552(top)
              Protocol next hop: 10.200.86.3
              Push 16
              Indirect next hop: 8dfd210 262148
              State: <Secondary Active Int Ext>
              Local AS:  7176 Peer AS:  7176
              Age: 3d 21:32:59       Metric2: 3
              Task: BGP_7176.10.200.86.3+65455
              Announcement bits (2): 0-BGP RT Background 1-KRT
              AS path: I
              Communities: target:100:200
              Import Accepted
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.3
              Primary Routing Table bgp.l3vpn.0
```

This output shows us:

- Dalwhinnie advertises the 192.168.90.12/30 route to CE2.
- The dalwhinnie-to-oban LSP is dalwhinnie's next-hop for this route in the aspen.inet.0 table.

- The selected path is the primary path for the dalwhinnie-to-oban LSP (the other path shown is the failover path for the LSP).
- Two labels are added to the packet upon egress to the core: 16 and 301328 (the label stack has two labels). 301328 is the top label in the stack.

In order to understand the forwarding plane, the reader must understand how to use the mpls.0 table, which is a list of all the labels known to the local router and the next hop associated with the label.

Since 301328 is the top label in the stack in the preceding output, it will be the label acted upon by the next-hop router. The output above also shows that the packet will egress dalwhinnie via the ge-0/0/2.0 interface, which leads to glenlivet. Let's check:

```
ps@glenlivet> show route table mpls.0 label 301328 detail

mpls.0: 44 destinations, 44 routes (44 active, 0 holddown, 0 hidden)
301328 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router
              Next-hop reference count: 1
              Next hop: 192.168.86.9 via ge-0/0/1.0 weight 0x1, selected
              Label-switched-path dalwhinnie-to-oban
              Label operation: Swap 301248
              Next hop: 192.168.86.45 via ge-0/0/3.0 weight 0x8001
              Label-switched-path Bypass->192.168.86.9
              Label operation: Swap 301248, Push 299920(top)
              State: <Active Int>
              Local AS:  7176
              Age: 6d 22:34:58       Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I
```

And the mpls.0 table on glenlivet shows that when it receives a packet with label 301328, it takes the following actions:

- Swap the 301328 label for the 301248 label.
- Send the packet along the dalwhinnie-to-oban LSP.
- Route the packet out interface ge-0/0/1.0 to the blair router.

Looking at the mpls.0 table on blair shows that upon receipt of a packet with the label 301248, it will pop (remove) the label and route the packet out ge-6/0/0.0 to oban along the *dalwhinnie-to-oban* LSP:

```
ps@blair> show route table mpls.0 label 301248 detail

mpls.0: 52 destinations, 52 routes (52 active, 0 holddown, 0 hidden)
          . . . <truncated for brevity> . . .
301248 (S=0) (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router
              Next-hop reference count: 1
              Next hop: 192.168.86.25 via ge-6/0/0.0 weight 0x1, selected
              Label-switched-path dalwhinnie-to-oban
              Label operation: Pop
              Next hop: 192.168.86.17 via ge-0/0/3.0 weight 0x8001
              Label-switched-path Bypass->192.168.86.25
              Label operation: Swap 299920
              State: <Active Int>
              Local AS:  7176
```

```
            Age: 6d 22:39:40      Metric: 1
            Task: RSVP
            Announcement bits (1): 0-KRT
            AS path: I
```

NOTE The `(S=0)` entry in the mpls.0 table refers to a packet coming into the router with a label stack depth of n>=2 exiting the router with a label stack depth of n-1. The information without the S=0 label refers to a packet entering the router with a label stack depth of 1 and exiting the router with no label. In the example above, the packet enters the router with a stack depth n=2.

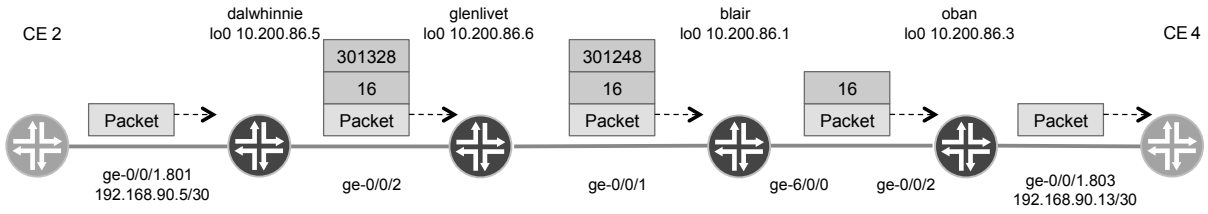
When the packet arrives at oban, it only has one label on the stack, with a value of 16. The output below shows that when oban receives a label with the value of 16, it pops the label and forwards it to the *aspen.inet.0* table for routing lookup.

```
ps@oban> show route table mpls.0 label 16

mpls.0: 32 destinations, 32 routes (32 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

16                 *[VPN/0] 4d 17:56:27
                      to table aspen.inet.0, Pop

ps@oban>
```

Once in the *aspen.inet.0* table, the packet gets routed out the ge-0/0/1.803 interface toward CE4 (the label exchange in the PE and P routers is transparent to the users on CE2 and CE4):

```
ps@oban> show route 192.168.90.14 table aspen detail

aspen.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
192.168.90.12/30 (1 entry, 1 announced)
       *Direct Preference: 0
              Next hop type: Interface
              Next-hop reference count: 1
              Next hop: via ge-0/0/1.803, selected
              State: <Active Int>
              Age: 5d 18:11:12
              Task: IF
              Announcement bits (1): 1-BGP RT Background
              AS path: I
```

Figure 2.5 illustrates these described label operations starting with dalwhinnie and ending with oban.




CE interface names are not shown.

If only one interface name appears for a link, then that is the interface name for the routers on both sides of the link.

All interface logical units are .0 unless otherwise noted.

Figure 2.5 End-to-end Label Operations for L3VPN Service with RSVP

## Layer 2 MPLS VPN

Layer 2 MPLS virtual private networks (VPNs) provide a method to transfer Layer 2 data point-to-point across an MPLS network. The MPLS transport is transparent to the CE end users – from the perspective of two CEs connected by one of these circuits, the remote IP address appears to be directly connected. There are two types of Layer 2 VPNs: the type signaled by BGP (*l2vpn*), and the type signaled by LDP (*l2circuit*).

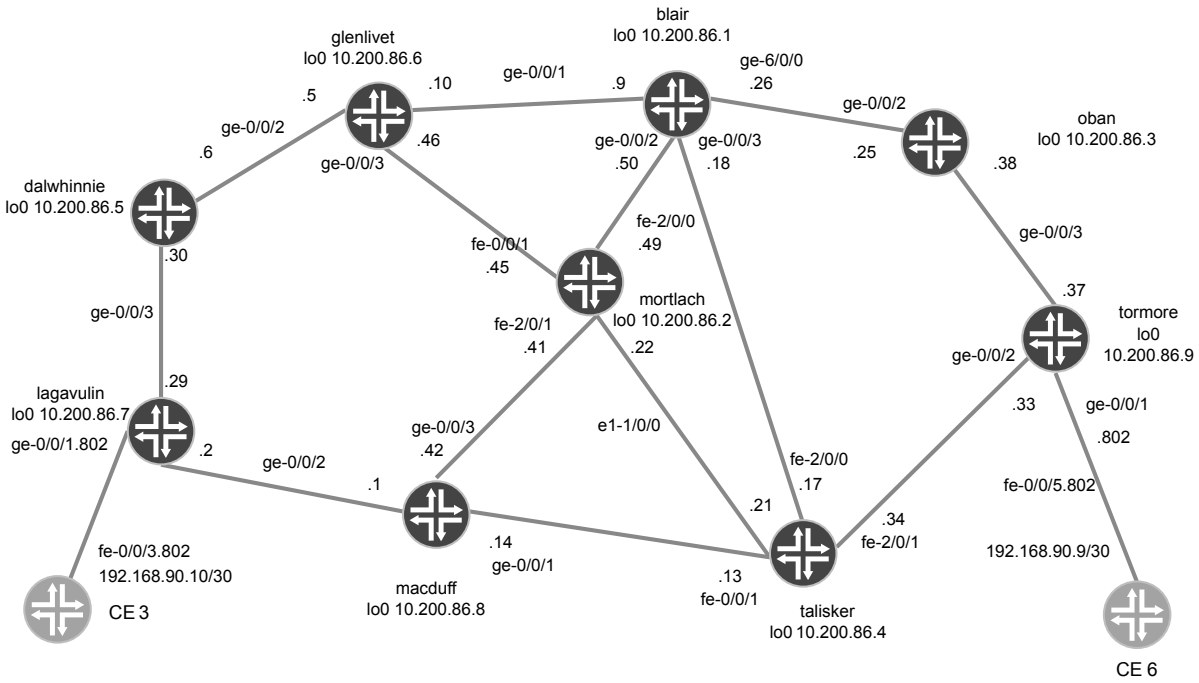
### L2VPN

An *l2vpn* connection (also known as a *Kompella* tunnel) is provisioned inside a routing-instance on two PE routers. The configuration for an l2vpn on lagavulin below shows many similarities to an l3vpn configuration:

- An interface that participates in the routing-instance
- A route-distinguisher
- A vrf-target

NOTE The l2vpn is based on draft-kompella-l2vpn-l2vpn-00.txt. For more information, see http://tools.ietf.org/html/draft-kompella-l2vpn-l2vpn-00.

These components have the same functionality as in an l3vpn. Figure 2.6 shows the layout of the l2vpn connection between the lagavulin and tormore routers.




All interface IP addresses are in /30 subnets in the 192.168.86/24 network unless otherwise noted.

All logical interface units are .0 unless otherwise noted.

If only one interface name appears for a link, then that is the interface name for the routers on both sides of the link.

Figure 2.6 Network Layout for L2VPN Connection

Additional information in the l2vpn routing-instance under the `[protocols l2vpn]` stanza includes the encapsulation-type (set for *ethernet-vlan* because the interface participating in the instance is using vlan-tagging) and the `[site]` stanza. Within the site stanza, the site-identifier is a 16-bit number greater than zero that uniquely identifies the site within the VPN and helps to control connectivity between sites. For example, the lagavulin instance's site-identifier of `3` corresponds to tormore's remote-site-id, while its remote-site-id of `6` corresponds to tormore's site-identifier value:

```
ps@lagavulin> show configuration routing-instances pine
instance-type l2vpn;
interface ge-0/0/1.802;
route-distinguisher 10.200.86.7:10;
vrf-target target:200:100;
protocols {
    l2vpn {
        encapsulation-type ethernet-vlan;
        site ce3 {
            site-identifier 3;
            interface ge-0/0/1.802 {
                remote-site-id 6;
            }
        }
    }
}
```

```
ps@tormore> show configuration routing-instances pine
instance-type l2vpn;
interface ge-0/0/1.802;
route-distinguisher 10.200.86.9:10;
vrf-target target:200:100;
protocols {
    l2vpn {
        encapsulation-type ethernet-vlan;
        site ce6 {
            site-identifier 6;
            interface ge-0/0/1.802 {
                remote-site-id 3;
            }
        }
    }
}
```

Each CE-facing interface participating in a l2vpn must have some specific configurations applied. The CE-facing physical interface and each of its logical interfaces should have the same *circuit cross-connect* (ccc) encapsulation type specified. Vlans 512-4094 are reserved for ccc encapsulation; vlan values 511 and below can be used as normal vlan-tagged interfaces if desired. The sample output from lagavulin below shows unit 802 configured to run in an l2vpn, while unit 400 can either participate in an l3vpn or in the master routing-instance. Ethernet-based interfaces require vlan-tagging if they are configured for multiple logical units.

```
ps@lagavulin> show configuration interfaces ge-0/0/1
vlan-tagging;
encapsulation vlan-ccc;
unit 400 {
    vlan-id 400;
    family inet {
        address 192.168.90.17/30;
```

```
    }
}
unit 802 {
    description "to CE3";
    encapsulation vlan-ccc;
    vlan-id 802;
}
```

Frame-relay, PPP, and ATM interfaces can also be configured to support ccc encapsulation and run in l2vpns. Using different encapsulation types on each end of an l2vpn connection requires the use of translation cross-connect (tcc) encapsulation.

MORE? For more information on ccc encapsulation on PE interfaces please see http://www.juniper.net/techpubs/en_US/junos10.3/topics/usage-guidelines/vpns-configuring-ccc-encapsulation-for-layer-2-vpns.html. As of this writing, Junos 10.3 is the most recent Junos version.

MORE? For more information on using tcc encapsulation on PE interfaces please see http://www.juniper.net/techpubs/en_US/junos10.3/information-products/topic-collections/config-guide-vpns/index.html?topic-33769.html. As of this writing, Junos 10.3 is the most recent Junos version.

### L2VPN Control Plane

BGP acts as the control plane for l2vpn, distributing the network layer reachability information (NLRI) for the l2vpn. In this example, there is a full mesh of iBGP connections between the PE routers. Like an l3vpn, BGP provides the control plane for this type of circuit. This means that the lagavulin and tormore routers must be able to exchange NLRI specific to an l2vpn. Lagavulin's BGP configuration shows how this is accomplished: via the *l2vpn* family, allowing the router to advertise and receive l2vpn signaling information via BGP:

```
ps@lagavulin> show configuration protocols bgp
family inet {
    any;
}
family inet-vpn {
    any;
}
family l2vpn {
    signaling;
}
group ibgp {
    type internal;
    local-address 10.200.86.7;
    . . . <snip> . . .
    neighbor 10.200.86.9;
}
```

A few simple commands can illustrate how the control plane works and also provide some methodology for troubleshooting should it become necessary, as shown here where tormore is advertising its l2vpn route to lagavulin:

```
ps@tormore> show route advertising-protocol bgp 10.200.86.7 table pine

pine.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
  10.200.86.9:10:6:3/96
*                     Self                    100        I
```

A quick check shows that lagavulin is indeed receiving l2vpn information via BGP from tormore. The RD configured in tormore's l2vpn instance is present, along with the BGP *route-target* extended community:

```
ps@lagavulin> show route receive-protocol bgp 10.200.86.9 table bgp.l2vpn.0 detail

bgp.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 10.200.86.9:10:6:3/96 (1 entry, 0 announced)
    Import Accepted
    Route Distinguisher: 10.200.86.9:10
    Label-base: 800000, range: 2, status-vector: 0x0
    Nexthop: 10.200.86.9
    Localpref: 100
    AS path: I
    Communities: target:200:100 Layer2-info: encaps:VLAN, control flags:Control-Word, mtu: 0, site
preference: 100
```

A final check shows that tormore's route is present in pine's routing table on lagavulin:

```
ps@lagavulin> show route receive-protocol bgp 10.200.86.9 table pine

pine.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
  10.200.86.9:10:6:3/96
*                     10.200.86.9              100        I
```

Examining the entire route table for the pine l2vpn instance on lagavulin shows the entry generated from the local interface and the route received via BGP from tormore. Additionally, this output gives some insight into the forwarding plane because it shows the next-hop LSPs for the route from tormore:

```
ps@lagavulin> show route table pine

pine.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.7:10:3:5/96
                *[L2VPN/170/-101] 2w4d 23:17:36, metric2 1
                   Indirect
10.200.86.9:10:6:3/96
                *[BGP/170] 00:20:07, localpref 100, from 10.200.86.9
                   AS path: I
                 > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore
                   to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore-v2
```

To verify the status of the l2vpn forwarding connection itself, use the `show l2vpn connections` command. This command provides a wealth of valuable output for each l2vpn, including the status of the connection(s), the remote PEs address(es), the local interface status, and the l2vpn's label information:

```
ps@lagavulin> show l2vpn connections extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid     NC -- interface encapsulation not CCC/TCC/VPLS
 . . .<truncated for brevity> . . .
PF -- Profile parse failure     PB -- Profile busy
RS -- remote site standby

Legend for interface status
Up -- operational
Dn -- down
```

```
Instance: pine
  Local site: ce3 (3)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    ge-0/0/1.802       6
    Label-base       Offset    Size  Range    Preference
    800000          5        2     2        100
      status-vector:  0
    connection-site          Type  St    Time last up         # Up trans
    6                      rmt   Up    Aug 24 04:50:31 2010         1
      Remote PE: 10.200.86.9, Negotiated control-word: Yes (Null)
      Incoming label: 800001, Outgoing label: 800000
      Local interface: ge-0/0/1.802, Status: Up, Encapsulation: VLAN
    Connection History:
       Aug 24 04:50:31 2010  status update timer
       Aug 24 04:50:30 2010  PE route changed
       Aug 24 04:50:30 2010  Out lbl Update                 800000
       Aug 24 04:50:30 2010  In lbl Update                  800001
       Aug 24 04:50:30 2010  loc intf up              ge-0/0/1.802
```

NOTE At first glance, the l2vpn routes can appear to be a bit tough to understand. In a nutshell, the first two parts (10.200.86.9:10 and :6, for example) are the configured RD of the router advertising the route and the site ID where the advertisement is coming from. The last part (:3) is the label-offset (this offset is generated automatically and is used internally to generate VPN labels). The /96 indicates that the netmask for the 96 bit address is all 1s. See http://tools.ietf.org/html/draft-kompella-l2vpn-l2vpn-00 for a more in-depth breakdown on each part of the l2vpn route.

### L2VPN Forwarding Plane

As with l3vpn, it is possible to examine the forwarding plane to understand the label operations for traffic in an l2vpn. In the preceding `show l2vpn connections` output from lagavulin, the outgoing label for the connection is shown to be 800000; this is the first label pushed onto the packet bound for tormore. Looking at the l2vpn instance's routing table, the traffic bound for the route coming from tormore is taking the lagavulin-to-tormore LSP:

```
ps@lagavulin> show route table pine

pine.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.7:10:3:5/96
                *[L2VPN/170/-101] 2w5d 22:06:26, metric2 1
                  Indirect
10.200.86.9:10:6:3/96
                *[BGP/170] 23:08:57, localpref 100, from 10.200.86.9
                  AS path: I
                 > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore
                  to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore-v2
```

Looking at the RSVP session for the LSP, the outgoing label is 302160:

```
ps@lagavulin> show rsvp session lsp name lagavulin-to-tormore detail
Ingress RSVP: 10 sessions

10.200.86.9
  From: 10.200.86.7, LSPstate: Up, ActiveRoute: 0
```

```
  LSPname: lagavulin-to-tormore, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 302160
  Resv style: 1 FF, Label in: -, Label out: 302160
  Time left:   -, Since: Thu Aug  5 23:50:59 2010
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 34752 protocol 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.1 (ge-0/0/2.0) 36871 pkts
  RESV rcvfrom: 192.168.86.1 (ge-0/0/2.0) 36848 pkts
  Explct route: 192.168.86.1 192.168.86.13 192.168.86.33
  Record route: <self> 192.168.86.1 192.168.86.13 192.168.86.33
Total 1 displayed, Up 1, Down 0
```

So the packet egresses tormore with a label stack of 800000 and 302160 (800000 is the bottom label on the stack).

Looking on the next router in the path, when macduff sees the 302160 label, it swaps it for 300144 before sending it out ge-0/0/1.0 on the *lagavulin-to-tormore* LSP toward talisker:

```
ps@macduff> show route table mpls.0 label 302160 detail

mpls.0: 35 destinations, 35 routes (35 active, 0 holddown, 0 hidden)
302160 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 561
              Next-hop reference count: 3
              Next hop: 192.168.86.13 via ge-0/0/1.0 weight 0x1, selected
              Label-switched-path lagavulin-to-tormore
              Label operation: Swap 300144
```

When the talisker router sees the 300144 label, it performs a pop operation and sends the packet out on the *lagavulin-to-tormore* LSP egressing via the fe-2/0/1.0 interface:

```
ps@talisker> show route table mpls.0 label 300144 detail

mpls.0: 43 destinations, 43 routes (43 active, 0 holddown, 0 hidden)
          . . . <truncated for brevity> . . .
300144 (S=0)(1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 592
              Next-hop reference count: 2
              Next hop: 192.168.86.33 via fe-2/0/1.0 weight 0x1, selected
              Label-switched-path lagavulin-to-tormore
              Label operation: Pop
             . . . <truncated for brevity> . . .
```

Since talisker (the penultimate router) popped the top label, the label stack on the packet only contains the 800000 label. Tormore, the ultimate router, reads the label and assigns the packet to the pine routing-instance based on the label's value. Talisker pops the remaining 800000 label and sends it out via ge-0/0/1.802, toward the destination, *CE6*:

```
ps@tormore> show route table mpls.0 label 800000

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

```
800000             *[L2VPN/7] 23:59:23
                   > via ge-0/0/1.802, Pop      Offset: 4
```

Figure 2.7 illustrates these l2vpn forwarding plane operations.

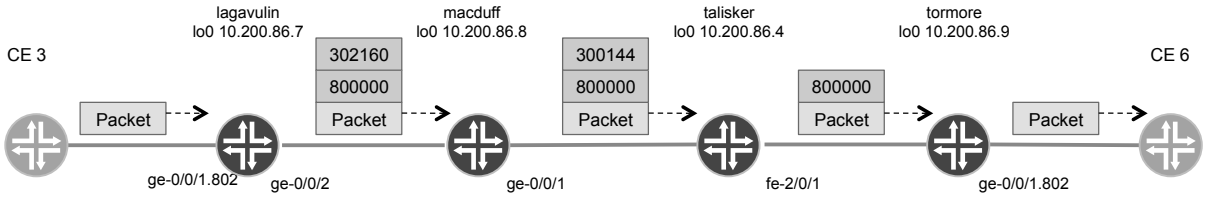



Figure 2.7 L2VPN End-to-end Label Operations

## l2circuits

Another type of Layer 2 VPNs, known as *l2circuits* in Junos, are sometimes referred to as *Martini tunnels*. They are another method to encapsulate Layer 2 protocols and transport them across an MPLS network.

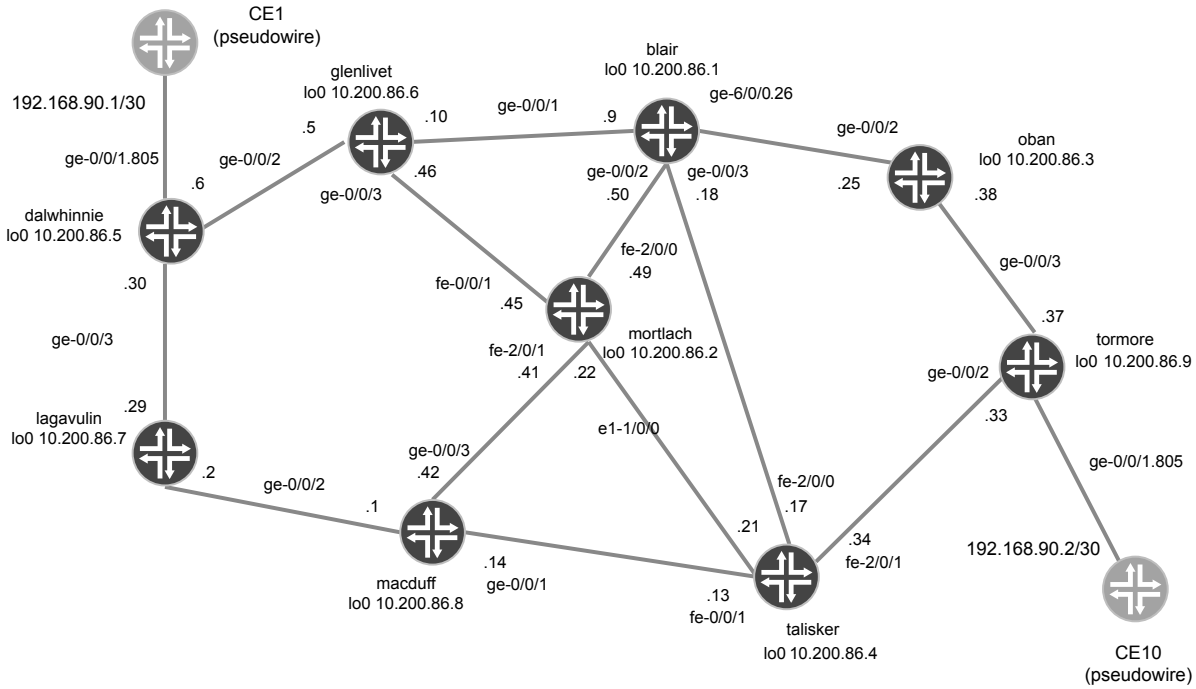
NOTE Layer 2 circuits are based on IETF RFC 4447. That document is available at http://www.rfc-editor.org/rfc/rfc4447.txt.

An l2circuit, like an l2vpn, can transport SONET, ATM, ethernet, or frame-relay encapsulated traffic in a mode transparent to the end user; as far as the CE end users at each end of an l2circuit can tell, they are directly connected. However, unlike l2vpns, l2circuits use LDP for signaling (instead of BGP), are not configured within a routing-instance, and do not use site-identifiers or route distinguishers. A standard l2circuit cannot establish connections between CEs that connect to the same PE router. However, if this local layer2 interface switching is desired, local-switching can be configured at the [edit protocols l2circuit] level. For example:

```
[edit]
ps@dalwhinnie# show protocols l2circuit local-switching
interface ge-0/0/1.200 {
  end-interface {
    interface ge-0/0/3.200;
  }
}
```

NOTE Since this book focuses on MPLS and MPLS services, further study of local-switching is beyond its scope.

Figure 2.8 shows the network layout and illustrates an l2circuit connection between dalwhinnie and tormore.




All interface IP addresses are in /30 subnets in the 192.168.86/24 network unless otherwise noted.

All logical interface units are .0 unless otherwise noted .

If only one interface name appears for a link, then that is the interface name for the routers on both sides of the link.

Figure 2.8 Network Layout for l2circuit Connection

In Figure 2.8, the CE-facing physical interface on the PE and each of its logical interfaces participating in the l2circuit have the same configuration as an interface participating in an l2vpn. The physical and logical interfaces on each endpoint PE router should have the same circuit cross-connect (ccc) encapsulation type specified. Again, vlans 512-4094 are reserved for ccc encapsulation; vlan values 511 and below can be used as normal vlan-tagged interfaces if desired. The following snapshot from dalwhinnie shows the physical interface ge-0/0/1 and its logical unit 805 configured for vlan-ccc encapsulation, while unit 403, with a vlan-id of 403, can participate in a l3vpn or in the master routing instance for normal IPv4 traffic. In addition to the PE's core-facing interfaces running LDP, the PE endpoint routers must also run LDP on their lo0 address. The remote PE's lo0 address must be configured under the `[edit protocols l2circuit]` stanza, along with the local interface participating in the Martini tunnel; on dalwhinnie:

```
ps@dalwhinnie> show configuration interfaces ge-0/0/1
vlan-tagging;
encapsulation vlan-ccc;
unit 403 {
    description "to CE2";
    vlan-id 403;
    family inet {
        address 192.168.90.5/30;
    }
}
unit 805 {
    description "to CE1";
    encapsulation vlan-ccc;
```

```
   vlan-id 805;
}

ps@dalwhinnie> show configuration protocols ldp
interface ge-0/0/2.0;
interface ge-0/0/3.0;
interface lo0.0;

ps@dalwhinnie> show configuration protocols l2circuit
neighbor 10.200.86.9 {
   interface ge-0/0/1.805 {
      virtual-circuit-id 10;
   }
}
```

And just to be thorough, here are the corresponding configurations on tormore:

```
ps@tormore> show configuration interfaces ge-0/0/1
vlan-tagging;
encapsulation vlan-ccc;
unit 802 {
   description "to CE6";
   encapsulation vlan-ccc;
   vlan-id 802;
}
unit 805 {
   description "to ce10";
   encapsulation vlan-ccc;
   vlan-id 805;
}

ps@tormore> show configuration protocols ldp
interface ge-0/0/2.0;
interface ge-0/0/3.0;
interface lo0.0;

ps@tormore> show configuration protocols l2circuit
neighbor 10.200.86.5 {
   interface ge-0/0/1.805 {
      virtual-circuit-id 10;
   }
}
```

With this configuration on both sides, the l2circuit comes up:

```
ps@tormore> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid     NP -- interface h/w not present
. . .<truncated for brevity> . . .
RD -- remote site signaled down  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 10.200.86.5
   Interface              Type  St    Time last up         # Up trans
   ge-0/0/1.805(vc 10)      rmt   Up    Aug 31 08:50:10 2010         4
     Remote PE: 10.200.86.5, Negotiated control-word: Yes (Null)
     Incoming label: 300000, Outgoing label: 301136
     Negotiated PW status TLV: No
     Local interface: ge-0/0/1.805, Status: Up, Encapsulation: VLAN
```

The l2circuit is up, using local interface ge-0/0/1.805 on tormore, the remote end is 10.200.86.5 (dalwhinnie), and the capability to use a control-word has been negotiated, but a control-word has not been set. Tormore expects traffic bound for ge-0/0/1.805 to have an incoming label 300000 and it sends traffic for this l2circuit to dalwhinnie with a label of 301136.

NOTE Standard behavior for Martini tunnels requires the same encapsulation type and VLAN-ID on both sides. Junos does allow non-standard behavior in the form of mismatching VLAN-IDs on each side of a Martini tunnel and different encapsulation types on each side – for example, ethernet on one side and vlan on the other.

### L2circuit Control Plane

LDP provides the control plane for l2circuits. Tormore's LDP neighbors include 10.200.86.5 (dalwhinnie), via tormore's lo0.0 interface. This connection is made possible via dalwhinnie's lo0 interface running LDP and tormore's targeted LDP session towards it, specified using the `neighbor` statement in the `[edit protocols l2circuit]` stanza:

```
ps@tormore> show configuration protocols l2circuit
neighbor 10.200.86.5 {
   interface ge-0/0/1.805 {
      virtual-circuit-id 10;
   }
}

ps@tormore> show ldp neighbor
Address          Interface        Label space ID       Hold time
192.168.86.34     ge-0/0/2.0       10.200.86.4:0          11
192.168.86.38     ge-0/0/3.0       10.200.86.3:0          14
10.200.86.5       lo0.0            10.200.86.5:0          37
```

Examination of the LDP routes available in the l2circuit.0 table on tormore shows one route coming from 10.200.86.5 (dalwhinnie):

```
ps@tormore> show route protocol ldp table l2circuit.0 extensive

l2circuit.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
10.200.86.5:CtrlWord:4:10:Remote/96 (1 entry, 1 announced)
      *LDP   Preference: 9
             Next hop type: Discard
             Next-hop reference count: 1
             State: <Active Int>
             Local AS:  7176
             Age: 6d 7:24:08
             Task: LDP
             Announcement bits (1): 1-l2 circuit
             AS path: I
             VC Label 301136, MTU 1500, VLAN ID 805
```

The l2circuit.0 output from dalwhinnie confirms the following:

- the route is being learned via LDP
- tormore must attach label 301136 for traffic to this route
- the l2circuit MTU is 1500 bytes
- the l2circuit vlan value is 805

The route format for an l2circuit route is `neighbor-address:control-word-status:encapsulation-type:vc-id:source`. Table 2.1 explains what each of these fields mean.

Table 2.1 l2circuit Route Format Fields

| Field | Meaning |
|---|---|
| `neighbor-address` | Address of the LDP neighbor advertising the route. |
| `control-word-status` | Shows whether the use of a control-word has been negotiated for the circuit. If a control-word-status has been negotiated, that does not mean that the circuit necessarily requires use of a control-word; it means that the circuit can use one if one is configured. |
| `encapsulation-type` | 1 - Frame Relay DLCI<br>2 - ATM AAL5 VCC transport<br>3 - ATM transparent cell transport<br>4 - Ethernet VLAN<br>5 - Ethernet<br>6 - HDLC<br>7 - PPP<br>9 - ATM VCC cell transport<br>10 - ATM VPC cell transport |
| `vc-id` | Virtual-circuit identifier. |
| `source` | Source of advertisement: either local or remote. |

### L2circuit Forwarding Plane

A Martini tunnel's forwarding plane can be illustrated with a few commands. On tormore, the *l2circuit.0* table is a good place to start to understand the label information for the forwarding plane back to dalwhinnie. The following output from the `show route table l2circuit.0 detail` command has two parts.

```
ps@tormore> show route table l2circuit.0 detail

l2circuit.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
10.200.86.5:CtrlWord:4:10:Local/96 (1 entry, 1 announced)
      *L2CKT  Preference: 7
              Next hop type: Indirect
              Next-hop reference count: 1
              Next hop type: Router
              Next hop: 192.168.86.34 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path tormore-to-dalwhinnie
              Label operation: Push 300592
              Protocol next hop: 10.200.86.5
              Indirect next hop: 8e50300 -
              State: <Active Int>
              Local AS:  7176
              Age: 1d 1:25:03       Metric2: 4
              Task: l2 circuit
              Announcement bits (1): 0-LDP
              AS path: I
              VC Label 300000, MTU 1500, VLAN ID 805

10.200.86.5:CtrlWord:4:10:Remote/96 (1 entry, 1 announced)
```

```
      *LDP   Preference: 9
             Next hop type: Discard
             Next-hop reference count: 1
             State: <Active Int>
             Local AS:  7176
             Age: 1w0d 8:45:57
             Task: LDP
             Announcement bits (1): 1-l2 circuit
             AS path: I
             VC Label 301136, MTU 1500, VLAN ID 805
```

Let's examine the second part of the output first. It shows the `10.200.86.5:CtrlWord:4:10:Remote/96` destination entry, received via LDP, and shows that tormore pushes a VC label 301136 on to the packet bound for that destination. The first part of the output shows a destination `10.200.86.5:CtrlWord:4:10:Local/96`, which is the same entry as before, but with a *local* source instead of *remote*. This switch in source shows that tormore has accepted that route into its local table and is ready to forward traffic to that destination. Tormore uses the *tormore-to-dalwhinnie* LSP to reach that destination, pushing label 300592 onto the packet to send it on that LSP.

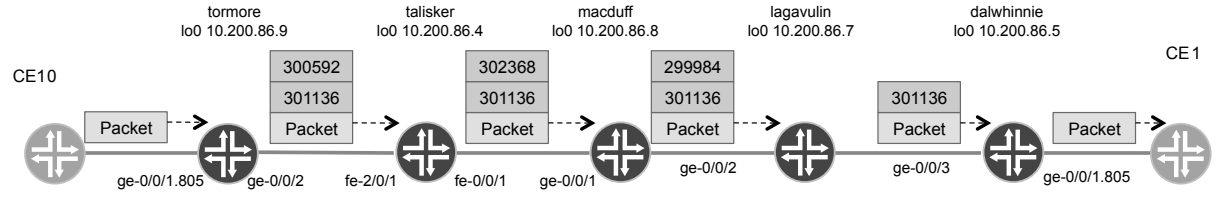
NOTE The *VC Label 300000* output beneath the L2CKT section means that tormore forwards packets that it receives with label 300000 out its local CE-facing interface.

NOTE This output is a typical style for when an RSVP LSP is available between the circuit endpoint routers. If the core is running LDP but not RSVP, the output would be similar, but would not have the *label-switched-path* line; the *label operation* line would still be present, but the label shown would be a label for an LDP LSP.

All this information serves as a starting point to map the forwarding label operations from tormore to dalwhinnie for this Martini tunnel circuit (recall how to see an RSVP LSP's path using the `show mpls lsp name <LSP-name> detail` command). The following outputs are from each next-hop router along the LSP, and create the basis for Figure 2.9, shown after the output examples:

```
ps@talisker> show route table mpls.0 label 300592 detail

mpls.0: 43 destinations, 43 routes (43 active, 0 holddown, 0 hidden)
300592 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 608
              Next-hop reference count: 3
              Next hop: 192.168.86.14 via fe-0/0/1.0 weight 0x1, selected
              Label-switched-path tormore-to-dalwhinnie
              Label operation: Swap 302368
. . . <truncated for brevity> . . .

ps@macduff> show route table mpls.0 label 302368 detail

mpls.0: 36 destinations, 36 routes (36 active, 0 holddown, 0 hidden)
302368 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 596
              Next-hop reference count: 3
              Next hop: 192.168.86.2 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path tormore-to-dalwhinnie
              Label operation: Swap 299984
. . . <truncated for brevity> . . .

ps@lagavulin> show route table mpls.0 label 299984 detail
```

```
mpls.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
         . . . <truncated for brevity> . . .
299984(S=0) (1 entry, 1 announced)
      *RSVP   Preference: 7
            Next hop type: Router, Next hop index: 607
            Next-hop reference count: 2
            Next hop: 192.168.86.30 via ge-0/0/3.0 weight 0x1, selected
            Label-switched-path tormore-to-dalwhinnie
            Label operation: Pop
. . . <truncated for brevity> . . .

ps@dalwhinnie> show route table mpls.0 label 301136 detail

mpls.0: 15 destinations, 15 routes (15 active, 0 holddown, 0 hidden)
301136 (1 entry, 1 announced)
      *L2CKT  Preference: 7
            Next hop type: Router, Next hop index: 571
            Next-hop reference count: 2
            Next hop: via ge-0/0/1.805, selected
            Label operation: Pop      Offset: 4
            State: <Active Int>
            Local AS:  7176
            Age: 1w0d 22:18:07
            Task: Common L2 VC
            Announcement bits (1): 0-KRT
            AS path: I
```




CE interface names are not shown.

If one interface name appears between two routers, the interface name on each router is the same on both ends of the link.

All interface logical units are .0 unless otherwise noted.

Figure 2.9 Example l2circuit End-to-end Forwarding Label Operations

## Martini and Kompella

Before moving on, it's important to note that Martini (*l2circuits* in Junos) and Kompella (*l2vpn* in Junos), have many similarities. For example, both create point-to-point Layer 2 connections and both use the same encapsulations (see Figure 2.10). But there are also some important differences between them as well. Martini uses LDP for signaling, while Kompella uses BGP. In small environments, Martini is advantageous because it is easier to configure and requires no existing BGP infrastructure. In larger networks, or networks likely to expand, Martini suffers from inherent scalability problems because each PE requires a targeted LDP session to every other PE (if the goal is to be able to provide point-to-point connectivity between each PE). Unlike Martini, Kompella does not suffer from this *Nx(N-1)* scalability problem because BGP provides the control plane, and BGP has tools that allow it to effectively scale. Also, in larger networks that would employ BGP route reflectors, Kompella's auto-discovery mechanism means less configuration and

fewer control-plane connections versus Martini, where each PE always requires a separate TCP session to the remote PE (multiple l2circuits between those two PEs can be signaled over that single TCP session, however). Finally, if the network planner plans to deploy a suite of MPLS services (L3VPN, Layer 2 VPN, VPLS), it makes sense to leverage the necessary BGP infrastructure to offer Kompella, which requires BGP, over Martini. Table 2.2 highlights some of these differences.

Table 2.2 Brief Martini/Kompella Comparison

| Aspect | Martini | Kompella |
|---|---|---|
| Ease of implementation on small networks | Relatively easy | Relatively complex |
| Ease of implementation in larger networks | Starts to require more configuration and router resources | Leverages BGP scalability as network grows |
| Scalability | Not very scalable | Very scalable |
| Signaling protocol | LDP | BGP |
| Leverages common resources used by other MPLS services | No | Yes |

Figure 2.10 shows how data encapsulation changes from when the packet arrives at the PE router on an ethernet VLAN to the PE encapsulating the data for transport over an l2circuit or l2vpn. L2circuits and l2vpns use the same type of encapsulation for transport over the core MPLS network.

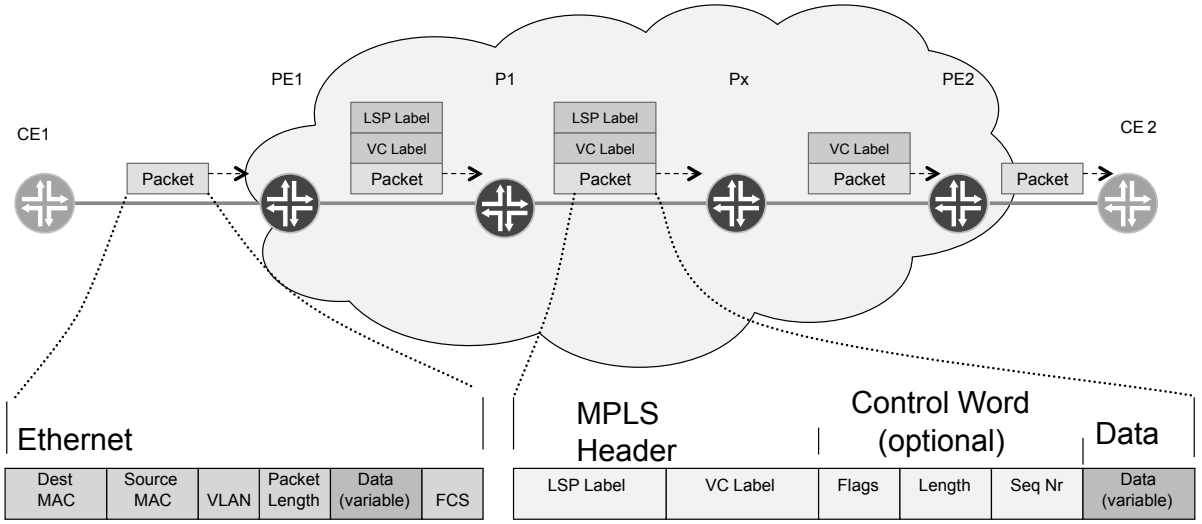



Figure 2.10 Encapsulation for l2circuit and l2vpn Packets

WARNING It is important to understand that there is a Martini *encapsulation*, which describes how layer 2 data is *encapsulated* when traversing an l2circuit or l2vpn and a Martini *tunnel*, which describes how the circuit is *signaled*. Martini tunnels (l2circuits) and

Kompella tunnels (l2vpn) use different signaling protocols (LDP and BGP, respectively), but use a common encapsulation (Martini encapsulation). In general, Martini tunnels have wider vendor support, so they are typically preferred for interop between different vendor PEs.

## VPLS

VPLS stands for *Virtual Private LAN Service*; it is a virtual LAN service provided by a network's PE routers. From the perspective of the CE router, VPLS looks and acts like a single LAN. While it is possible to use a series of Martini or Kompella tunnels to create behavior that mimics a broadcast domain, doing so requires manual mapping of a PE's local circuits to each remote PE; and even that exercise does not give point-to-multipoint capabilities. This mapping exercise can be labor-intensive, time-consuming, and confusing, especially when the goal is to create a full mesh between PE routers. VPLS removes this burden, allowing broadcast domain behavior without the manual site mapping. In fact, with the advent of VPLS, it may be preferable for an enterprise to establish VPLS initially between the first two sites, and then add additional sites as needed, to provide a seamless transition from Layer 2 point-to-point to Layer 2 multi-site broadcast domain behavior.

There are two standards for VPLS: RFC 4761, which uses BGP for signaling, and RFC 4762, which uses LDP for signaling. RFC 4761 (BGP) has shown itself to be preferable in most situations for the following reasons:

- LDP signaled VPLS requires a full-mesh of LDP sessions between PE routers; if a new router is added to the mesh, it requires provisioning the targeted LDP session to that router on every other router in the mesh.
- LDP requires manual provisioning of each circuit (via the neighbor statement within the VPLS instance); BGP signaling has native auto-discovery.

NOTE Very recent developments (as of this writing) in LDP-VPLS do allow auto-discovery using BGP. This does simplify provisioning, but still adds the extra protocol (LDP) to support VPLS. In contrast, BGP-VPLS uses BGP for both signaling and auto-discovery.

- BGP provides the signaling framework to provide all types of MPLS services (VPLS, layer 2 VPN, and layer 3 VPN) in a scalable framework.
- Hierarchical VPLS, a method to make LDP signaled VPLS more scalable, requires additional capital in the form of layer 2 switches; BGP-signaled VPLS can scale without the additional layer 2 switches.

While this list is not comprehensive, it does highlight some key points as to why BGP is preferable over LDP for VPLS signaling. So with that pointed out, this section focuses on the BGP implementation for VPLS. (If you need information on how to configure Junos devices for LDP signaled VPLS, see http://www.juniper.net/techpubs/en_US/junos10.1/topics/usage-guidelines/vpns-configuring-vpls-routing-instances.html#id-vpls-ldp-signal .)

Figure 2.11 illustrates the layout of the network and the CEs involved in the VPLS instance *oak*.

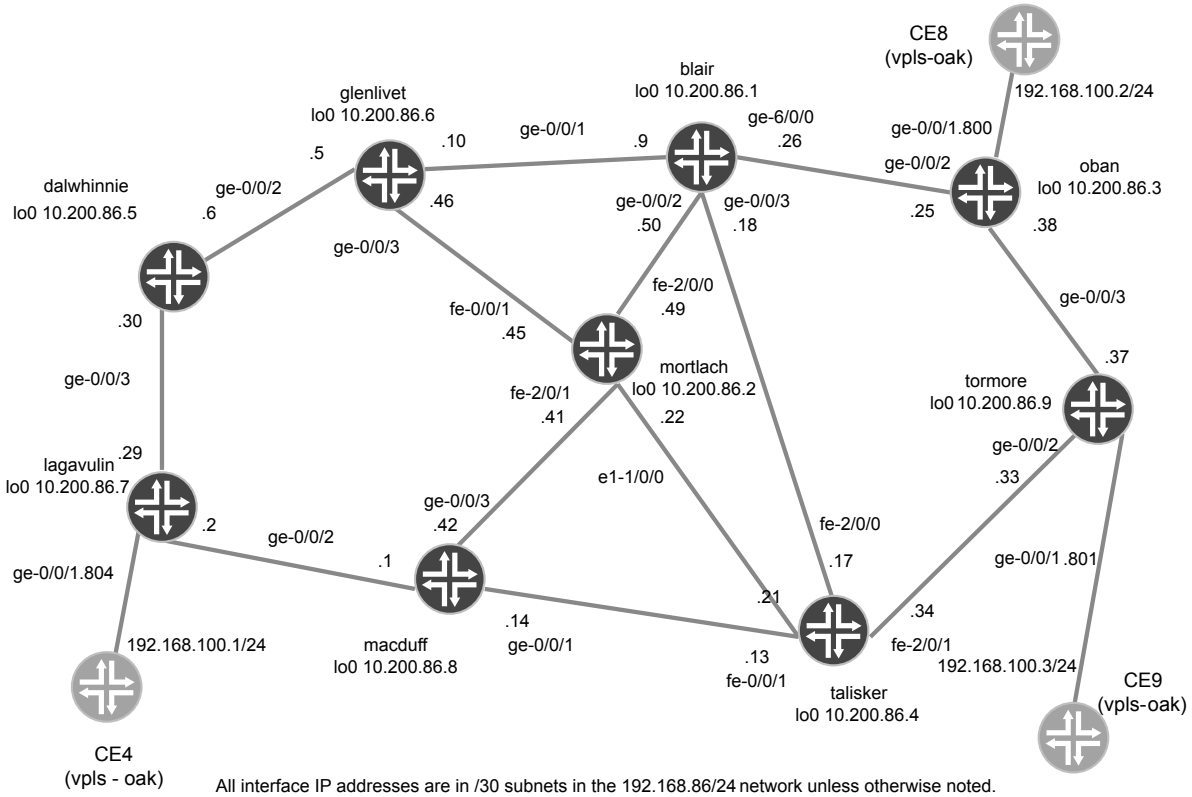



Figure 2.11 VPLS Layout

As is the case with l2vpn, VPLS requires BGP to have the family `l2vpn` with `signaling` in order to advertise the VPLS NLRI information:

```
family l2vpn {
   signaling;
}
```

As shown in Figure 2.11, lagavulin's CE-facing interface shows the specific configurations necessary for an interface to participate in a VPLS instance. The physical interface must be set for `vlan-tagging` and `vlan-vpls` encapsulation. The logical interface participating in the instance must be set for encapsulation `vlan-vpls`. This configuration also allows for separate logical interfaces on ge-0/0/1 that can run IPv4 and can participate in a layer 3 VPN or in the master routing-instance for internet access:

```
ps@lagavulin> show configuration interfaces ge-0/0/1
vlan-tagging;
encapsulation vlan-vpls;
unit 400 {
   description "Internet connection";
   vlan-id 400;
   family inet {
      address 192.168.90.17/30;
   }
}
unit 804 {
```

```
    description "VPLS to CE4";
    encapsulation vlan-vpls;
    vlan-id 804;
    input-vlan-map pop;
    output-vlan-map push;
    family vpls;
}
```

NOTE Initially all the P and PE routers in the testbed (all J-series) were running Junos 9.6R4.4. Juniper's release notes for Junos 9.5 (http://www.juniper.net/techpubs/en_US/junos9.5/information-products/topic-collections/release-notes/9.5/j-series-new-features.html) state that VPLS support begins in this load. However, attempting to configure VPLS on the J2350s resulted in a warning that VPLS is not supported (see below). The routers running VPLS (coincidentally all were J2350s) were upgraded to 10.0R3.10 and were then able to run VPLS

```
[edit routing-instances test]
ps@lagavulin# show
##
## Warning: statement ignored: unsupported platform (j2350)
##
instance-type vpls;

[edit routing-instances test]
ps@lagavulin# run show version
Hostname: lagavulin
Model: j2350
JUNOS Software Release [9.6R4.4]
```

Of note are the *vlan-maps* configured on unit 804 as shown in Figure 2.12. On most Juniper routers, vlan-maps allow communication within a VPLS instance to different CE sites, with each CE site able to have a different vlan-id value.

NOTE The MX series routers do not have to use vlan-maps. See Appendix A for some MX-specific VPLS configurations and capabilities.

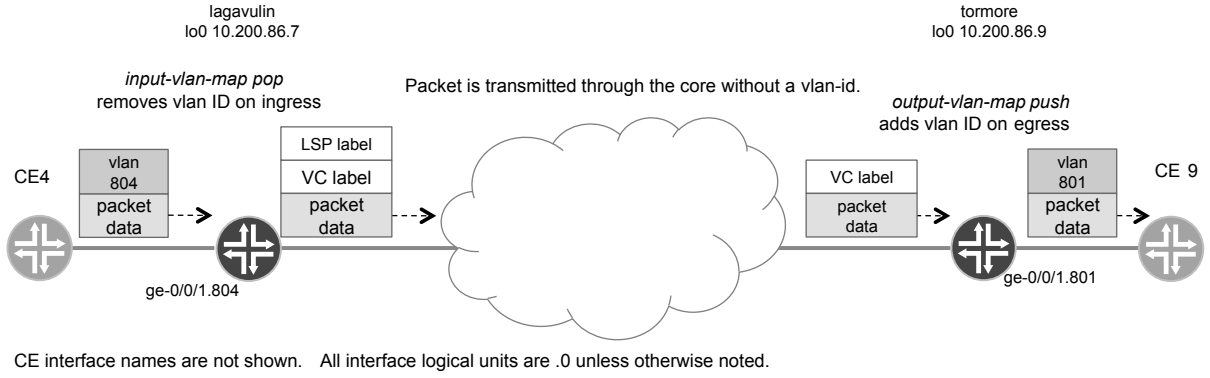



Figure 2.12 VPLS Vlan-map Example

The mechanics of the vlan-maps in Figure 2.12 shows the packet leaving router CE4 and arriving at lagavulin with a vlan-id of 804. The `input-vlan-map pop` on lagavulin's ge-0/0/1.804 interface *pops* (similar to a label *pop*) the vlan-id tag from the packet. The packet traverses the core, arriving at tormore. Upon egress from tormore, the `output-vlan-map push` on ge-0/0/1.801 *pushes* (similar to a label *push*) the 801 vlan-id tag on to the packet and sends the packet to CE9. The `output-vlan-`

`map push` function will automatically add the logical interface's configured vlan-id to the packet unless a different vlan-id is explicitly specified. If vlan-ids are the same at every site in a given VPLS instance, then vlan-maps are not necessary.

The actual VPLS routing-instance configuration contains the familiar RD and RT, along with the interface(s) participating in the instance. Similar to l2vpns, VPLS instances also have unique site-identifiers specified for each CE site in the common VPLS instance:

```
ps@lagavulin> show configuration routing-instances oak
instance-type vpls;
interface ge-0/0/1.804;
route-distinguisher 10.200.86.7:100;
vrf-target target:300:200;
protocols {
   vpls {
      site-range 5;
      no-tunnel-services;
      site ce4 {
         site-identifier 1;
         interface ge-0/0/1.804; ## matches interface to specific site
      }
   }
}
```

For thoroughness and future reference, here are tormore's and oban's *oak* VPLS routing-instance configurations:

```
ps@tormore> show configuration routing-instances oak
instance-type vpls;
interface ge-0/0/1.801;
route-distinguisher 10.200.86.9:100;
vrf-target target:300:200;
protocols {
   vpls {
      site-range 5;
      no-tunnel-services;
      site ce9 {
         site-identifier 3;
         interface ge-0/0/1.801; ## matches interface to specific site
      }
   }
}

ps@oban> show configuration routing-instances oak
instance-type vpls;
interface ge-0/0/1.800;
route-distinguisher 10.200.86.3:100;
vrf-target target:300:200;
protocols {
   vpls {
      site-range 5;
      no-tunnel-services;
      site ce8 {
         site-identifier 2;
         interface ge-0/0/1.800; ## matches interface to specific site
      }
   }
}
```

The *no-tunnel-services* configuration allows VPLS to function on a J-series router without the use of an additional tunnel-services PIC. This configuration creates an *lsi*

interface on the local router for each remote site in the VPLS instance. The output from `show vpls connections` on lagavulin shows that the connection to site 2 is down because oban is reporting that its ce-facing interface is down. The connection to site 3 (tormore) is up and receiving packets on local interface lsi.1050113.

```
ps@lagavulin> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid     NC -- interface encapsulation not CCC/TCC/VPLS
 . . . <snipped for brevity> . . .
RD -- remote site signaled down  SC -- local and remote site ID collision
. . . <snipped for brevity> . . .
PF -- Profile parse failure     PB -- Profile busy
RS -- remote site standby

Legend for interface status
Up -- operational
Dn -- down

Instance: oak
  Local site: ce4 (1)
    connection-site         Type  St     Time last up          # Up trans
    2                       rmt   RD
    3                       rmt   Up     Sep  1 07:29:50 2010         1
      Remote PE: 10.200.86.9, Negotiated control-word: No
      Incoming label: 262147, Outgoing label: 262145
      Local interface: lsi.1050113, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls oak local site 1 remote site 3
```

## VPLS Control Plane

For the RFC 4761 version of VPLS, BGP performs the control plane functions, handling the advertisements between PE routers. The control plane for BGP signaled VPLS performs two primary functions: auto-discovery, and setup and teardown of the layer 2 connections that comprise the VPLS domain. In BGP-signaled VPLS, each PE has the capability to automatically discover which other PEs share common VPLS instances and then signal a layer 2 connection to that PE. In turn, it also has the capability to discover when a PE is no longer a member of a common VPLS instance and automatically removes the layer 2 connection to that remote PE. Compare this to an LDP-signaled, configuration-based approach, which can become configuration and planning intensive as the network grows and evolves. BGP requires the `family l2vpn signaling` configuration within BGP to advertise the layer 2 NLRI to neighbor PEs running VPLS; the VPLS NLRI format is the same as draft-Kompella (l2vpn).

Lagavulin is receiving one route each from tormore and oban in the *oak* routing table; the RDs and RT received can be verified by examining the previously shown VPLS routing-instance configurations for those routers:

```
ps@lagavulin> show route receive-protocol bgp 10.200.86.9 detail table oak

oak.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
* 10.200.86.9:100:3:1/96 (1 entry, 1 announced)
     Import Accepted
     Route Distinguisher: 10.200.86.9:100
     Label-base: 262145, range: 8
     Nexthop: 10.200.86.9
     Localpref: 100
```

```
    AS path: I
    Communities: target:300:200 Layer2-info: encaps:VPLS, control flags:, mtu: 0, site preference:
100

ps@lagavulin> show route receive-protocol bgp 10.200.86.3 detail table oak

oak.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
* 10.200.86.3:100:2:1/96 (1 entry, 1 announced)
    Import Accepted
    Route Distinguisher: 10.200.86.3:100
    Label-base: 262145, range: 8
    Nexthop: 10.200.86.3
    Localpref: 0
    AS path: I
    Communities: target:300:200 Layer2-info: encaps:VPLS, control flags:Site-Down, mtu: 0, site
preference: 100
```

As expected, both routes have the RT BGP extended community value of *target:300:200*, allowing the *oak* instance to import that route. The RD values for each route differ, as expected, keeping each route unique, even in the event of a possible duplication of routes if other VPLS instances are present.

Since VPLS performs Layer 2 switching, at this point the reader may be asking the obvious question: *How do I tell which MAC addresses are in the VPLS domain?*

The MAC addresses for each VPLS domain are stored in each instance's forwarding table; viewing this table can be invaluable for troubleshooting. Lagavulin shows two MAC addresses in its *oak* forwarding table. The first-listed MAC is reachable out of interface ge-0/0/2.0, with a next-hop of 192.168.86.1 when the router performs the two listed *push* label operations. The second listed MAC is the directly connected CE: the egress interface listed is ge-0/0/1.804, which is the CE-facing interface that participates in the *oak* VPLS routing-instance.

```
ps@lagavulin> show route forwarding-table vpn oak
Routing table: oak.vpls
VPLS:
Destination       Type RtRef Next hop         Type Index NhRef Netif
default          perm    0                  rjct   565    1
lsi.1050113       user    0                  comp   613    2
ge-0/0/1.804      user    0                  comp   605    2
00:24:dc:df:07:05/48 dynm    0                  indr 262150    4
                         192.168.86.1     Push 262145, Push 302160(top)   614   1 ge-0/0/2.0
00:26:88:03:1c:03/48 dynm    0                  ucst   603    3 ge-0/0/1.804
```

## VPLS Forwarding Plane

The *oak* forwarding-table lets us see how the local router forwards packets destined for remote VPLS destinations. Lagavulin's *oak* forwarding-table shows *Push 262145, Push 302160(top)* as the next-hop type to reach the *00:24:dc:df:07:05/48* MAC address, with the top label being the one that the next hop router acts on. A quick search of RSVP sessions for the 302160 label tells us that packets bound for that MAC address use the *lagavulin-to-tormore* LSP:

```
ps@lagavulin> show rsvp session | match 302160
10.200.86.9    10.200.86.7    Up      0  1 FF      -   302160 lagavulin-to-tormore
```

CAUTION The next-hop label-switched-path for the specific MAC address in the VPLS forwarding-table may not match the listed next-hop label-switched-path for the NLRI in the route table if there are multiple, equal-cost LSPs between the ingress and egress PE routers. Notice that the output below shows the label-switched-path *lagavulin-to-tormore-v2* (instead of *lagavulin-to-tormore* in the preceding output) as the active path for the NLRI coming from tormore. In the case of equal-cost paths, Junos uses a hashing algorithm to determine which path to use on a per-destination basis. In the case of VPLS, the hash determines the specific equal-cost LSP to reach a given MAC:

```
ps@lagavulin> show route table oak

oak.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

. . . <snipped for brevity> . . .
10.200.86.9:100:3:1/96
                *[BGP/170] 1w5d 20:37:32, localpref 100, from 10.200.86.9
                  AS path: I
                  to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore
                 > to 192.168.86.1 via ge-0/0/2.0, label-switched-path lagavulin-to-tormore-v2
```

## Try It Yourself: Map VPLS Label Operations

Okay, enough telling you what happens. It's time for you to tell yourself. Using the sample output below, create a label operations illustration like the previous ones featured for L3VPN and L2VPN for the *oak* VPLS instance.

```
ps@macduff> show route table mpls.0 label-switched-path lagavulin-to-tormore detail

mpls.0: 45 destinations, 45 routes (45 active, 0 holddown, 0 hidden)
302160 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 561
              Next-hop reference count: 4
              Next hop: 192.168.86.13 via ge-0/0/1.0 weight 0x1, selected
              Label-switched-path lagavulin-to-tormore
              Label operation: Swap 300144
              State: <Active Int AckRequest>
              Local AS:  7176
              Age: 5w4d 4:11:09      Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I

ps@talisker> show route table mpls.0 label-switched-path lagavulin-to-tormore detail


mpls.0: 43 destinations, 43 routes (43 active, 0 holddown, 0 hidden)
300144 (1 entry, 1 announced)
       *RSVP   Preference: 7
              Next hop type: Router, Next hop index: 592
              Next-hop reference count: 4
              Next hop: 192.168.86.33 via fe-2/0/1.0 weight 0x1, selected
              Label-switched-path lagavulin-to-tormore
              Label operation: Pop
              State: <Active Int>
              Local AS:  7176
              Age: 5w4d 4:14:23      Metric: 1
              Task: RSVP
```

```
            Announcement bits (1): 0-KRT
            AS path: I

ps@tormore> show route table mpls.0 source-gateway 10.200.86.7

mpls.0: 19 destinations, 19 routes (19 active, 0 holddown, 0 hidden)

ps@tormore> show route table mpls.0 label 262145

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

262145             *[VPLS/7] 1w5d 20:36:43
                   > via lsi.1049600, Pop
```

## Summary

This chapter not only explained L3VPN, Layer 2 VPNs, and VPLS, but provided some basic configurations for them, too. It also highlighted the roles of the route-target and route-distinguisher and the different functions for each – and again, it is crucial for the reader to understand the difference in those roles in order to form a basis for basic understanding of MPLS services with a BGP control plane.

Additionally, this chapter showed that BGP-controlled MPLS services have more inherent scaling ability and provide more synergies than LDP-controlled options. The reader should now understand how to configure basic configurations for MPLS services as well, and have a good understanding of how to troubleshoot those services. If not, you might reread those sections before moving on to Chapter 3, where this book shows you how to best design a network based on *your* specific constraints and requirements.

# Chapter 3

## MPLS Core Implementations

It's time to build on the concepts and principles revealed in the first two chapters and present options for building and scaling your MPLS network. This chapter covers physical and logical architecture options, device roles, as well as scaling options. The goal is to not only to provide guidance for safely building out your network, but also to help you begin to think about the hurdles your growing network might face. Let's begin with a few definitions and an analysis of the MPLS network at hand, then consider the different requirements of the network and translates that into design options. After that, scaling considerations are detailed and potential solutions and mitigating steps are proposed.

## Purpose of Network

When designing an MPLS network, it's important to identify and understand the purpose of the network. Will the network support performance sensitive traffic? Will resiliency be a primary factor? What services (such as MPLS VPNs) must the network support? Is class-of-service a requirement? The answers to these questions, and others like them, are critical in understanding how to design a successful network and will help you identify the requirements of your network in three major categories: performance requirements, resiliency requirements, and service requirements.

- Performance requirements: These are, admittedly, a very broad category, but the most commonly considered performance requirements covered in this book are speed (latency), differences in latency (jitter), bandwidth guarantees, prioritization (class-of-service), and class-based forwarding. Understanding how the network needs to behave to meet these performance requirements will assist you in the deployment of an IGP, and potentially MPLS metric schemes, class-of-service classification, and scheduling parameters, and will likely play a major role in your traffic-engineering decisions.
- Resiliency requirements: *Resiliency* is the ability of a network to adapt to a topology change. Link down events are a great example of a topology change (a router down event is simply a super set of a link down event), but link up events are just as important. How resilient a network needs to be has a significant impact on your network design, specifically affecting timers, IGP and MPLS resiliency features, and potentially class-of-service. Resiliency requirements and service requirements are tightly coupled concepts. The question comes down to balancing resiliency and complexity.

NOTE Complexity is a commonly underestimated consequence of network design. A network is only as good as the operator's ability to manage it.

- Service requirements: Understanding the service requirements of your network is perhaps the most important aspect of network design. A network required to pass inet.0 routed IP packets with no regard for performance, resiliency, or differentiated services yields a simple design. An IGP, and potentially BGP, is all it takes to meet these needs. However, if the services offered require differentiation and guarantees, class-of-service, and traffic engineering, a bandwidth reservation protocol may become necessary. Delivering real-time traffic such as voice, video, and gaming to your network will undoubtedly require quicker convergence times and force the engineer to consider the IGP and MPLS resiliency options. Offering any type of VPN (Layer-3 VPN, Layer-2 VPN, VPLS) requires additional consideration and for the most part necessitates the use of MPLS. It is also important to note that these requirements are not

exclusive. Offering MPLS based VPN services clearly necessitates some sort of MPLS deployment. This will likely change the current class-of-service implementation, or force the creation of one. It may also force the engineer to investigate traffic engineering where before it was not a concern. The addition of a voice-service offering may also affect class-of-service and inspire a resiliency overhaul. All of these issues point to an overarching principle: understanding a network's service requirements is paramount to the success of that network.

- The services offered on a network are always changing. Services are modified, extended, removed, and added as fast as there is a profit to be made and this book in no way suggests that doing it right in the beginning guarantees long-term success. Building a flexible and adaptive network is what is important. The discussions and examples provided throughout this book seek to help you build just that: a scalable, flexible, and most importantly, manageable MPLS network.

## Router Roles

Defining the roles and responsibilities of a router provides a road map for the configuration and requirements of that device. For our purposes, this discussion focuses on the Provider (P) router and the Provider Edge (PE) router. P routers are usually core only devices – these routers will run an IGP (and may or may not run BGP, even if the edge routers do!) – and primarily forward switch labels (rather than IP packets). The PE device will participate in the IGP, may or may not run BGP (depending on the network), and switch IP packets and forward IP packets into MPLS LSPs.

There are two common role conventions in an MPLS environment, a strict P/PE separation, and a hybrid model in which devices act as P *and* PE routers.

### Separated Provider (P) and Provider Edge (PE) Design

In the separate P and PE design, the P routers forward MPLS labels rather than IP packets. The MPLS label is the forwarding differentiator, not the destination address in the IP header. The PE routers maintain all internal and external routing information (local, IGP, and potentially BGP routes), and perform routing lookups based on destination addresses. They then prepend the IP packets with an MPLS label and forward the label as appropriate. Because of this, the P routers only receive labels (rather than IP packets), and do all of their forwarding based on this label information. The P routers will perform a label lookup to find the outgoing label mapped to the received label, and swap the top label.

One major advantage of this model, illustrated in Figure 3.1, is that the P routers only forward based on labels, so they only need IGP routes and MPLS label information. Networks that utilize BGP do not need to run BGP on the P routers, reducing the resource utilization on these devices and dramatically reducing the size of the internal BGP full-mesh. Additionally, there are some operational benefits to having a hard demarcation between the core and edge. Many organizations like to separate the edge and core to either optimize hardware and software testing and deployment, or separate responsibility and access between different teams, or both. A consequence of this model is that P routers cannot terminate any classical *edge* service.

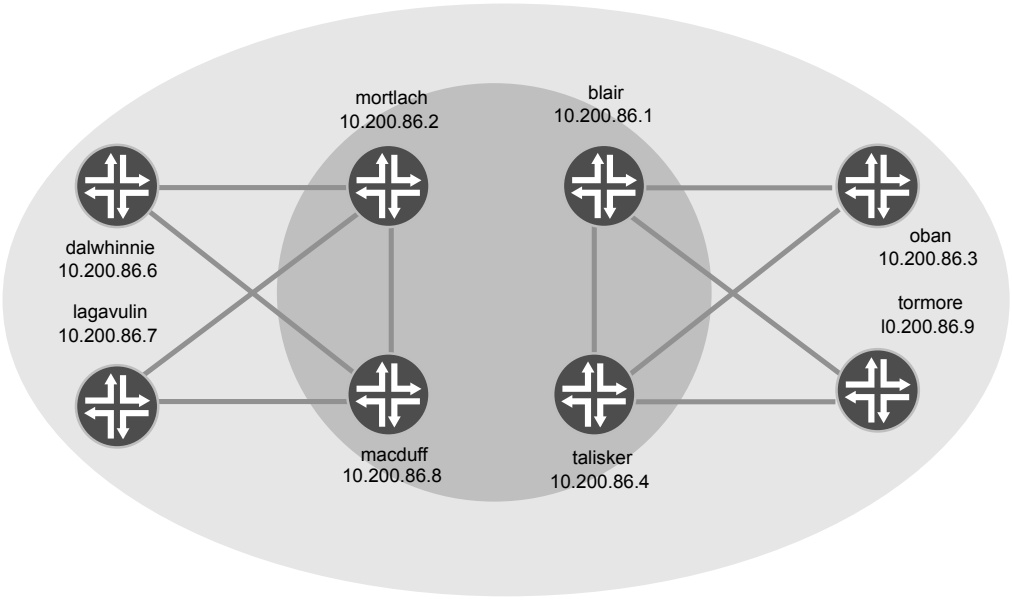



Figure 3.1 Separated Core/edge Design

A significant advantage to the separated core/edge model is that the separation makes operations and outage scope more deterministic as services are limited to the edge. Feature requirements and upgrade cycles also benefit from the separated model, as do access limitations that can limit the impact of user errors.

## Hybrid Provider (P) and Provider Edge (PE) Design

An alternative to the strict P/PE separation model is a hybrid model in which routers serve as both Provider and Provider Edge devices. In this model, what would traditionally be P routers in the separated model shown in Figure 3.1, also perform some edge functionality. Whether this is terminating customers, connecting to external BGP peers, or providing any other service, it may require that the router have additional routing information. The result is that the PE device will forward based on the IP destination for some amount of traffic. The PE devices may also provide transit paths to other routers on the network, which means that for this traffic, it will switch based on label information. This makes it a P router as well. The advantage to this model is the flexibility of providing services from any device, and the downside is that more devices will need a more complete routing table.

## P/PE Design Selection

You may be wondering which design is right for you? The answer is largely dependent upon the physical connectivity of your network.

Network operators are at the mercy of the network because the physical topology is often already in place, or at a minimum, is driven more by available budget than by protocol design. Frequently, two other criteria help answer P/PE design questions: edge-to-core physical connectivity and the willingness of the business to leave core routers as just that – in other words, no edge connectivity on core devices.

There are strong arguments for a separated edge and core design – an exclusive edge places the complexity at the edge, leaving the core to focus on switching packets fast. It also provides an operational demarcation, allowing for differentiated, group-based authorization levels based on device role. Code requirements and testing, too, can be limited to a smaller set of hardware and roles, which means testing and deployment groups can focus on different features for different groups of devices. Finally, limiting edge services and connections to edge devices makes it easier to identify the scope of planned and unplanned outages, reduce the impact of core changes, and can assist in capacity planning.

One of things that might help you define your P/PE design is how you connect at the edge. Let's examine this for a moment.

### Edge Connectivity

With regard to edge connectivity, there are two common designs: the *U* (or *box)* design and the *X* design. The naming for these designs comes from the letters their design forms when viewed as a diagram. (Note there are also hybrid versions of these designs, but the economics and decision-making factors make them very similar.)

In the U design, paired edge routers each have an uplink to the core routers, with an edge-router cross-connect for redundancy. Figure 3.2 illustrates the U design.

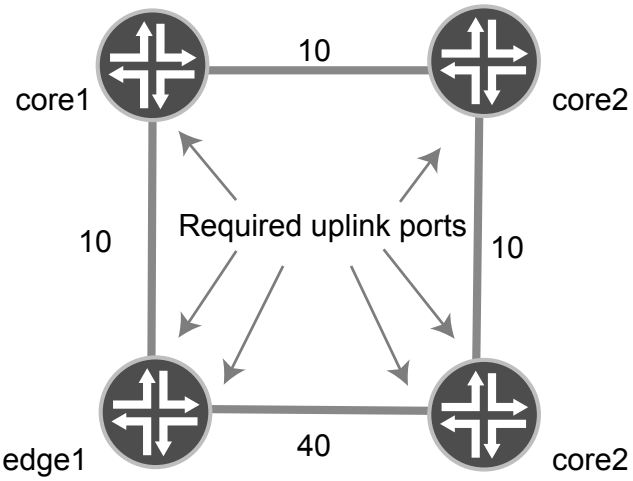



Figure 3.2 An U Edge Design

The advantage to the U design is cost. Redundant uplinks for two edge devices require only three circuits (6 ports), one from edge1 to core1, one from edge2 to core2, and one between the edge routers. The downside is oversubscription, because two edge devices share an uplink in a failure condition. Increasing the size of the uplinks mitigates this risk, through either discreet bandwidth increase or bundling, but this likely reduces the cost benefit. The pairing of edge devices can also lead to unfair, preferential treatment of traffic between paired edge routers, as intra-pair traffic need not transit the core routers. Setting IGP metrics appropriately can prevent this behavior (metrics that accomplish this goal are included in Figure 3.2). The *X* design reverses this trade-off, increasing redundant capacity at the cost of expense. Figure 3.3 illustrates this topology.

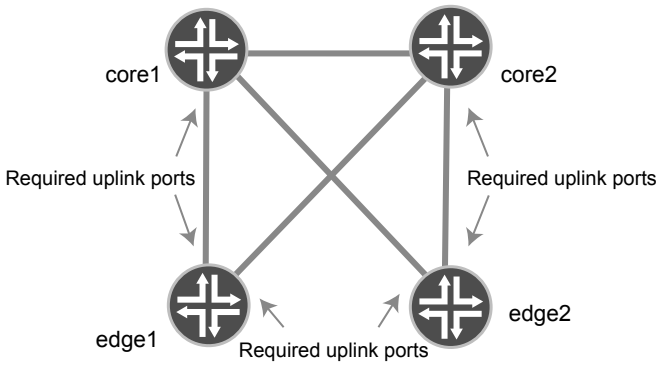



Figure 3.3 The X Edge Design

In the X edge design, full-bandwidth redundancy increases costs, but reduces the potential for outages and service impacts. Each edge router requires two uplinks (one uplink to each of the core routers), or four ports total, but if sized appropriately, uplink circuits should never be over utilized.

NOTE Core ports are a finite sum quantity and enough edge routers can consume all of these ports. A distribution layer may be necessary in these cases, as shown in Figure 3.4.

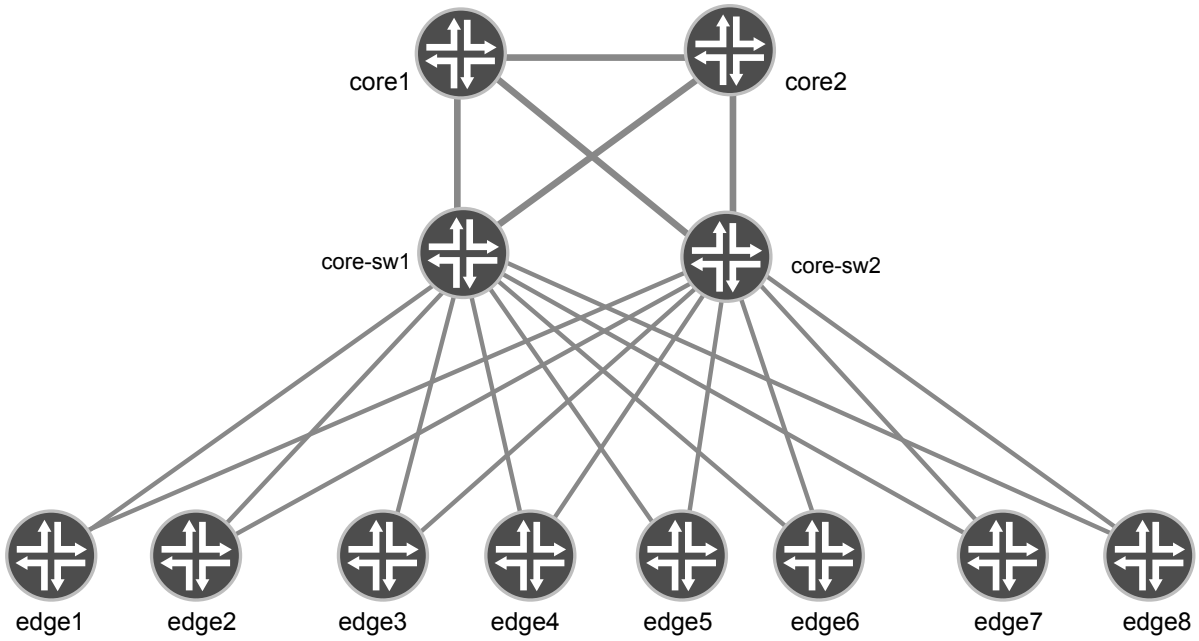



Figure 3.4 X Design with Aggregation Layer

The *X* design supports the separated P/PE model, as transit traffic will not traverse a PE router (with appropriate IGP metrics). The *U* design effectively makes every PE a P as well, as traffic may transit a PE router in the event of an uplink failure.

Finally, the other piece to this puzzle is whether edge routers exclusively serve edge services. If traditional edge services are limited to edge routers only, the network can

support a separated P and PE network. If core ports provide edge services, then every router in the network has the potential to be a PE and the network does not support role separation in this way. Table 3.1 lists the possible combinations and P/PE outcomes.

Table 3.1 P/PE Design Decision Matrix

| Physical Topology | Service Separation Commitment | P/PE Outcome |
|---|---|---|
| U Design | Inconsistent | Hybrid |
| X Design | Inconsistent | Hybrid |
| X Design | Total | Separated P/PE |

## Core Label Distribution

If your network will support MPLS, the implementer must select a label distribution protocol. The two options are the Label Distribution Protocol (LDP) and the Resource Reservation Protocol (RSVP). Your answers to the basic questions regarding the purpose of the network at the very beginning of this chapter, should help make this decision.

For environments that only require MPLS services such as RFC 4364-style Layer 3 VPNs, Kompella, or Martini-style Layer 2 pseudo-wires or VPLS, but do not require MPLS traffic-engineering or improved resiliency (over standard IGP convergence), LDP may be the right protocol. (Note that RSVP's complexity may not be worth the benefits in this case.) On the other hand, RSVP *is required* if fast-failover, MPLS traffic engineering, point-to-multipoint LSPs, or any of several other features are needed. Essentially the requirements of the network and the acceptable complexity of your network will help make this decision. LDP is extremely simple, but not feature-rich. RSVP has many features and benefits, but requires significantly more architectural engineering and operational effort. RSVP might require scaling techniques depending on the size of the network and number of LSPs, while LDP does not have this limitation.

Why not both you say? Hybrid MPLS designs are actually somewhat common and can help mitigate some RSVP scaling limitations. Many networks use RSVP as their primary distribution protocol, but will also configure LDP as a backup protocol due to its ease of implementation and operation.

### LDP in the Core

The major advantage to running LDP (only) in the core is its simplicity. LDP is not a routing protocol; it is purely a label distribution protocol. This means it simply advertises labels for routes in the routing-table and the standard route selection rules still apply (longest match, protocol preference, and protocol independent selection criteria). The downside to LDP is also its upside feature of simplicity – LDP does not currently support traffic engineering and fast-failover.

### RSVP in the Core

Because RSVP is so feature-rich, it is a common choice for within the core of large networks (especially service-providers) to provide label-switched-paths across a network with traffic engineering and bandwidth constraints, improved resiliency, and other features such as point-to-multipoint LSPs for use with RFC 4364- style Layer 3 VPNs and VPLS. Major factors that must be considered when deploying an RSVP-TE MPLS network include the number of routers participating in RSVP (and as a consequence, the number of LSPs that will be created), failover improvements and how they will add to the number of LSPs, traffic-engineering policies, service implications, and the often overlooked factor of operational supportability. All of the available tools within RSVP-TE can provide an extremely efficient, full-featured (with regard to services), and quick-healing network, but these same tools can lead to a network that is difficult to understand and engineer and nearly impossible to troubleshoot.

## Core MPLS Designs

Now that you have a good view of network requirements, selected protocols, and identified resiliency characteristics, let's examine some of the different network designs, their components, and the advantages and disadvantages to consider about each. The first option covered is an LDP only network, which is followed by an RSVP full mesh architecture. After reviewing these, we will turn back to that hybrid MPLS design point made a few paragraphs ago and focus on an LDP edge with a RSVP core with an LDP-tunneling component.

### LDP-only Network

Routers configured for LDP form a neighborship with directly connected neighbors using UDP as a transport mechanism, as shown in the example of a small LDP network in Figure 3.5.

NOTE When using LDP, labels are bound to loopback addresses and exchanged with LDP neighbors. A Juniper router will create a label for its own loopback address (generally a label of 3), creates mappings for labels received from peers, and then advertises those local and mapped labels to LDP peers.

In Figure 3.5, each router has LDP labels for three destination prefixes – the loopback addresses of the other three routers. Two of these will be a label of 3, as each router is the penultimate hop for two routers and the other will be a randomly assigned label for the non-directly connected router. For example, mortlach is directly connected to blair as well as macduff and it should therefore have received a label of 3 from each. Talisker, however, is not directly connected and both blair and macduff should have created label mappings for the label received from talisker and advertised those labels to mortlach. Mortlach performs a standard IP route selection for talisker and selects the label associated with that route; in this case the path through blair (20) is cheaper than the path through macduff (30). The output from mortlach shows this information in the inet.3 MPLS table as well as the inet.0 routing table:

```
ps@mortlach> show route 10.200.86.4
inet.0: 13 destinations, 18 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.4/32     *[LDP/9] 00:00:04, metric 1    ß Selected route through blair (metric 20)
```

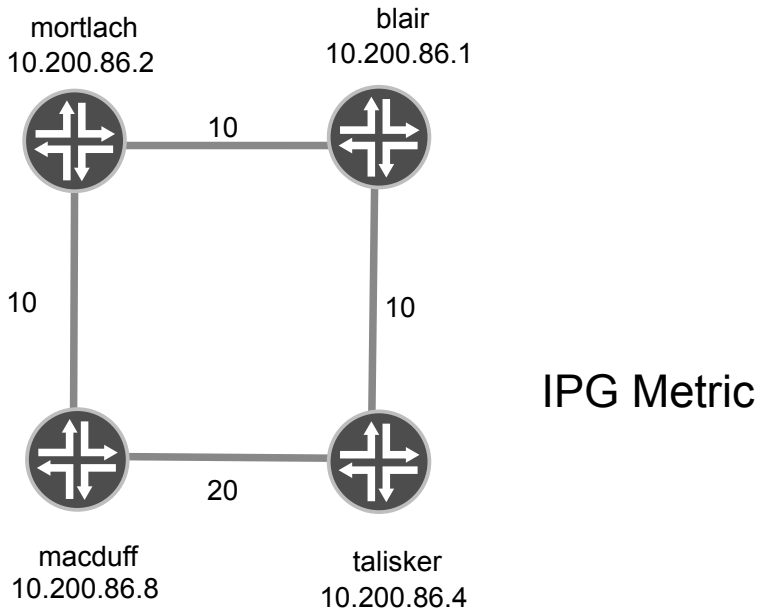



Figure 3.5 LDP Label Distribution Illustration

```
                > to 192.168.14.6 via ge-0/0/1.0, Push 100976
                [OSPF/10] 00:00:04, metric 20
to 192.168.14.6 via ge-0/0/1.0
```

Note that the LDP routes for blair and macduff do not show any label information, this is because mortlach received a label of 3 for these routes, indicating that mortlach is the penultimate hop. It therefore does not add a label to packets destined to either of these destinations, as shown here:

```
ps@mortlach> show route table inet.3
inet.3: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.1/32     *[LDP/9] 00:08:50, metric 1    ß LDP label for blair, no label
                 > to 192.168.14.6 via ge-0/0/1.0          shown (label 3)
10.200.86.4/32     *[LDP/9] 00:01:12, metric 1    ß LDP label for talisker, via blair
                 > to 192.168.14.6 via ge-0/0/1.0, Push 100976
10.200.86.8/32     *[LDP/9] 00:08:49, metric 1    ß LDP label for macduff, no label
                 > to 192.168.14.1 via ge-0/0/1.0          shown (label 3)

If blair were to receive a packet with a label of 100976, it would consult its mpls table to find the
label mapping for this label.
ps@blair> show route table mpls.0 label 100976
mpls.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100976             *[LDP/9] 00:27:08, metric 1
to 192.168.14.14 via ge-0/0/1.0, Pop
```

And here you can see that blair's operation is to pop the label (as it is the PHP router) and send it to talisker.

The important takeaway here is that LDP is not a routing protocol. It simply distributes labels and allows routers to associate them with selected routes. The IGP is still the determining factor, meaning that an LDP network can be used to transport mechanism for other protocols (or IP) for use with VPNs, but cannot be used for

other benefits associated with RSVP such as traffic-engineering and fast-reroute (LDP fast-reroute is not yet standardized as of the writing of this book). If traffic-engineering and improved resiliency are important aspects of your network, then LDP should not be used as the primary label distribution protocol (it can be used in conjunction with RSVP, which you'll see later).

NOTE If the main goal of an MPLS design is its fast resiliency benefits, an alternative approach is OSPF loop-free alternates (LFA). OSPF LFA provides fast failover without the MPLS overhead.

Hence, if the main goal of your network is to provide VPN services without any specific resiliency needs, the simplicity of an LDP implementation may be the right way for you to go.

## RSVP-TE

For those networks that require additional features, such as traffic-engineering and improved resiliency, RSVP is the right choice as the primary label distribution mechanism.

It is important to know that using RSVP comes with an operational and processing cost. While LDP is limited in its features, it is also extremely simple to configure and puts limited load on the network. LDP simply exchanges labels and routers use those labels for selected routes. Because RSVP affects routing decisions and builds paths across a network, routers must maintain state for those paths and must take more than just the routing table into consideration. This means routers must exchange additional network information, including available bandwidth and administrative group topology, and use it to build desired paths across the network. The routers use these messages to build and maintain state across the network. Additionally, routers need to refresh this information as the network topology changes. So clearly, there is more to RSVP than there is to LDP.

But with this complexity and additional processing comes functionality, and that may be critical to your environment. Traffic engineering allows administrators to send traffic over paths that may differ from what would be the shortest IGP path, which can provide a lower latency path, a less utilized path, or a method to route certain types of traffic around choke points. It can also prevent backup paths from transiting circuits that may run over the same physical path as primary paths. These are powerful features and RSVP is a great way to solve complex network problems.

NOTE LSP protection in any of its forms (Juniper fast-reroute, secondary LSPs, link-protection, and link-node protection) can offer network resiliency that cannot be attained through normal IP convergence, and this can be extremely valuable in environments that support fault-sensitive traffic such as voice and video.

## RSVP Edge-mesh Network

There are several different design options for RSVP-TE. One popular design extends RSVP-TE to the edge, providing full end-to-end support for traffic engineering and LSP protection. Note that it requires a single area/level IGP implementation, which may present scaling difficulties in large networks. But in our scenario, shown in Figure 3.6, the PE routers build LSPs to every other PE router, with P routers serving purely as transit label switching routers, and this works very well with the BGP-free core design discussed next.

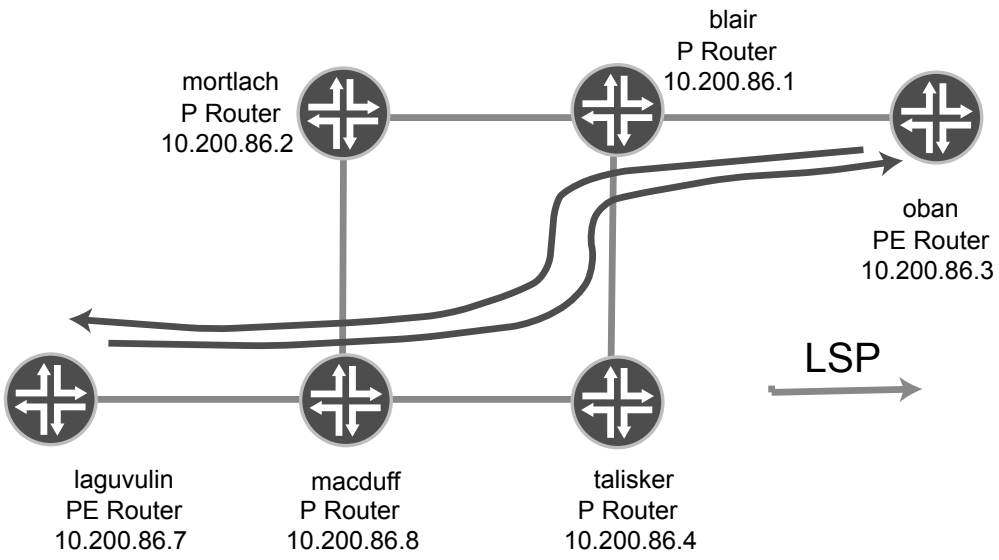



Figure 3.6 Edge LSP Mesh

## BGP-free Core

One advantage to a network in which all ingress packets are MPLS encapsulated on the edge is that the P routers no longer need a full routing table to switch packets. In traditional networks, routers forward traffic destined past the edge of the network based on an IGP or BGP, and each router has to have all of the routing information, or at least some portion of it, with the P routers nearly always requiring a full view of the network. On large networks with full Internet routing feeds, the full view could equate to hundreds of thousands of routes, requiring processing to build and maintain those peering relationships, memory to hold that information, and a mechanism to distribute this information to the forwarding plane (which also has to hold this information, by the way).

So if the remote PE's next-hop address can be associated with a label, the core P routers no longer need to know every route – instead, they only need to know every PE, and a label for that PE. Therefore, the ingress PE must encapsulate the incoming packet within an MPLS label, which gets the packet to the egress PE, removing the need for BGP routes on the core routers. Voila! The design reduces the control load, frees up resources, and overall improves network convergence. Both LDP and RSVP-TE support the BGP-free core design and large networks frequently use it.

## RSVP Full-mesh Network

In some networks, most, or all, of the routers in the network, serve as PEs. If this is the case, a full-mesh of LSPs is required to provide any-to-any MPLS connectivity with RSVP features. The situation can provide additional scaling issues because not only does the network operator have to consider the size of their OSPF area 0, or IS-IS Level 2 network, the number of LSPs created can become a burden.

For a network with n routers, a full-mesh creates `n(n-1)` LSPs, because each router must have an LSP to each other router. In a network with only 20 routers, this leads

to 380 LSPs. Grow the network to 100 routers and the LSP count is up to 9900, and this best-case scenario is assuming two unidirectional LSPs between a given set of routers and that no resiliency mechanism is used. For example, secondary LSPs immediately double the total number of LSPs (as every LSP has a backup, secondary LSP). Because LSP messaging is soft (think "send it and forget it"), there is no way for a router to know if it has received all the messages sent by a peer and there is no way for the sender to know if its intended target has received all of the messages it sent. In a situation in which a transit router has thousands of LSPs and a link goes down causing thousands of RSVP messages, it can become problematic if the sender or receiver cannot keep up with the volume of messages. If this does happen, head-end LSPs may continue sending packets into an LSP that is no longer up, which ends up with dropped traffic. This, of course, gets resolved as soon as the IGP can re-converge and RSVP can re-signal. However, if one of the goals of an RSVP deployment was to improve resiliency, problems like these can be significant.

In many RSVP implementations, there are routers that require an RSVP full mesh with every other router in the group and the operational impact of this requirement can be prohibitive; PEs offering L3VPN or VPLS services are a typical example. This full-mesh requirement implies that each router in a group of N routers must have N-1 RSVP LSPs configured. With the addition of each router to the group, each of the other routers requires an LSP to the new router. RSVP auto-mesh (available starting in Junos 10.1) greatly reduces the required configuration overhead for this type of RSVP implementation.

RSVP auto-mesh automatically creates a point-to-point RSVP LSP to each router that:

- It receives an iBGP route from (can be an l3vpn route, VPLS route, or inet.0 route).
- Has a lo0 address that resides in a specific address range.

NOTE Lab testing showed that in order for local router to spawn the RSVP auto-mesh LSPs, there had to be a labeled route to the iBGP next-hop (the LSP's destination) in the inet.0 table. The testing also showed that once the LSPs were up, the labeled route to the destination did not have to be in inet.0 anymore. To create the labeled path in inet.0, we enabled the LDP protocol and configured traffic-engineering bgp-igp-both-ribs under [protocols mpls], which created a labeled path to the destination and added that destination to inet.0. After the auto-mesh LSPs established, deactivating the LDP and traffic-engineering bgp-igp-both-ribs did not cause the auto-mesh LSPs to be torn down. This was tested on an MX-80 router running Junos 10.3R1.9.Configure auto-mesh under [routing-options dynamic-tunnels]:

```
ps@dalwhinnie> show configuration routing-options dynamic-tunnels
dt-default {
    rsvp-te dt-default-1 {
        label-switched-path-template {
            default-template;
        }
        destination-networks {
            10.200.86.0/24;
        }
    }
}
```

The label-switched-path-template in this example is the default-template; destination-networks is the allowed address range for the LSP endpoints. Both configura-

tions are required to be specified. This example creates an LSP to any router that sends dalwhinnie an iBGP route and whose lo0 address falls within the 10.200.86/24 range.

This configuration creates a tunnel family listing within inet.3 that contains the specified network:

```
ps@dalwhinnie> show route table inet.3 protocol tunnel

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.0/24     *[Tunnel/300] 01:13:13
                   Tunnel
```

And the dynamic LSP is spawned:

```
ps@dalwhinnie> show rsvp session ingress
Ingress RSVP: 1 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.7    10.200.86.4    Up      1  1 FF      -   300000 10.200.86.7:dt-rsvp-dt-default
Total 1 displayed, Up 1, Down 0

ps@dalwhinnie> show rsvp session ingress detail
Ingress RSVP: 1 sessions

10.200.86.7
  From: 10.200.86.4, LSPstate: Up, ActiveRoute: 1
  LSPname: 10.200.86.7:dt-rsvp-dt-default, LSPpath: Primary
  LSPtype: Dynamic Configured
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 300000
  Resv style: 1 FF, Label in: -, Label out: 300000
  Time left:    -, Since: Tue Mar 29 21:50:57 2011
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 34969 protocol 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 192.168.86.6 (ge-0/0/2.0) 13 pkts
  RESV rcvfrom: 192.168.86.6 (ge-0/0/2.0) 12 pkts
  Explct route: 192.168.86.6 192.168.86.10
  Record route: <self> 192.168.86.6 192.168.86.10
Total 1 displayed, Up 1, Down 0
```

Dalwhinnie now has a dynamic RSVP LSP to 10.200.86.7 (oban). The LSP name has the following format: [destination]:dt-rsvp-[name of tunnel]. The *dt-rsvp* signifies the module that created the tunnel (dynamic tunnel) and the tunnel type (RSVP).

See the routes eligible to take the tunnel here:

```
ps@dalwhinnie> show route table inet.3 10.200.86.7 detail

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
10.200.86.7/32 (1 entry, 1 announced)
       State: <FlashAll>
       *RSVP   Preference: 7/3
              Next hop type: Router, Next hop index: 663
              Next-hop reference count: 10
              Next hop: 192.168.86.6 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path 10.200.86.7:dt-rsvp-dt-default
              Label operation: Push 300000
              State: <Active Int>
```

```
            Local AS:    1
            Age: 8:39      Metric: 2
            Task: RSVP
            Announcement bits (2): 1-Resolve tree 1 2-Resolve tree 3
            AS path: I

ps@dalwhinnie> show route next-hop 10.200.86.7

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

15.15.15.0/24      *[BGP/170] 00:59:38, localpref 100, from 10.200.86.7
                    AS path: I
                  > to 192.168.86.6 via ge-0/0/2.0, label-switched-path 10.200.86.7:dt-rsvp-dt-
default

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

mpls.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)

bgp.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

                       10.200.86.3:100:4:1/96
                    *[BGP/170] 00:35:29, localpref 100, from 10.200.86.7
                      AS path: I
                     > to 192.168.86.6 via ge-0/0/2.0, label-switched-path 10.200.86.7:dt-rsvp-dt-
default

oak.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.3:100:4:1/96
                    *[BGP/170] 00:09:22, localpref 100, from 10.200.86.7
                      AS path: I
                     > to 192.168.86.6 via ge-0/0/2.0, label-switched-path 10.200.86.7:dt-rsvp-dt-
default
```

The output shows one inet.0 route and one VPLS route received from 10.200.86.7 that are eligible to take the auto-mesh LSP.

If you wish to use an LSP with features not included in the default-template, Junos allows you to specify a customized LSP template. The example below specifies a customized template named *dynamic-template* that requests link-protection for the LSP:

```
ps@dalwhinnie> show configuration routing-options dynamic-tunnels
dt {
   rsvp-te dt-1 {
      label-switched-path-template {
         dynamic-template;
      }
      destination-networks {
         10.200.86.0/24;
      }
   }
}

ps@dalwhinnie> show configuration protocols mpls
traffic-engineering bgp-igp-both-ribs;
 . . .
 . . .
```

```
label-switched-path dynamic-template {
   template;
   link-protection;
}
interface ge-0/0/2.0;
interface ge-0/0/3.0;


ps@dalwhinnie> show mpls lsp ingress detail
Ingress LSP: 1 sessions

10.200.86.7
  From: 10.200.86.4, State: Up, ActiveRoute: 1, LSPname: 10.200.86.7:dt-rsvp-dt
  ActivePath:  (primary)
  Link protection desired
  LSPtype: Dynamic Configured
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                 State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 2)
 192.168.86.6 S 192.168.86.10 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
         10.200.86.2(flag=0x20) 192.168.86.6(Label=300048) 10.200.86.7(flag=0x20)
192.168.86.10(Label=3)
Total 1 displayed, Up 1, Down 0
```

The auto-mesh LSP now has link-protection requested.

Finally, Junos does not allow an auto-mesh LSP to exist if there is not at least one iBGP route eligible to use it. If this condition occurs, the dynamic LSP starts a 15 minute countdown timer; if no route(s) become eligible for the LSP within this timeframe, Junos tears down the LSP. The example below shows a new configured LSP to the same destination as the auto-mesh LSP.

```
ps@dalwhinnie> show configuration protocols mpls
traffic-engineering bgp-igp-both-ribs;
label-switched-path dalwhinnie-to-oban {
   to 10.200.86.7;
   link-protection;
}


ps@dalwhinnie> show mpls lsp ingress
Ingress LSP: 2 sessions
To              From            State Rt P    ActivePath      LSPname
10.200.86.7     10.200.86.4     Up    0 *                     10.200.86.7:dt-rsvp-dt
10.200.86.7     10.200.86.4     Up    2 *                     dalwhinnie-to-oban
Total 2 displayed, Up 2, Down 0


ps@dalwhinnie> show dynamic-tunnels database
Table: inet.3

Destination-network: 10.200.86.0/24
Tunnel to: 10.200.86.7/32 (expires in 00:14:17 seconds)
  Reference count: 0
  Next-hop type: rsvp-te
    10.200.86.7:dt-rsvp-dt
```

In the case of both a configured LSP and auto-mesh LSP to the same destination, by default Junos prefers the configured LSP. In this example above, since the iBGP

routes prefer the configured LSP, no routes are eligible for the auto-mesh LSP, and the expiration countdown timer commences.

RSVP auto-mesh can be a valuable tool in scaling an RSVP implementation. It is important to ensure that your Junos version supports RSVP auto-mesh in conjunction with any desired HA options such as graceful routing-engine switchover (GRES) and non-stop routing (NSR).

A full-mesh of LSPs can certainly work on small networks that will not grow, but on large or growing networks, the scaling issues can become problematic. Fear not though; there are options available to get similar functionality without risking the stability of your network.

## Hybrid LDP Edge and RSVP Core Network

One option for gaining edge-to-edge MPLS functionality without paying the full-mesh price is to run RSVP only between core (P) devices while extending MPLS to the edge with LDP. Luckily, RSVP-TE supports this concept by enabling the tunneling of LDP packets over RSVP-signaled LSPs, as shown in Figure 3.7. This means that the ingress and egress routers of an LSP directly exchange LDP labels, effectively building a single hop LDP path between each label switch router. Through this method, edge-to-edge LDP signaled MPLS paths can establish over RSVP signaled LSPs in the core, providing a more scalable MPLS implementation while gaining the traffic-engineering and improved resiliency benefits of RSVP LSPs.

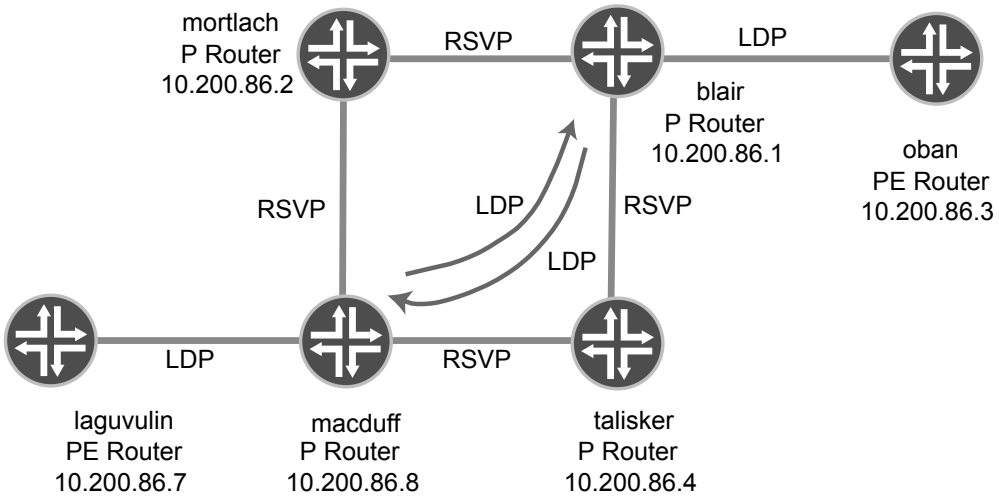



Figure 3.7 LDP Tunneling

In Figure 3.7, lagavulin will have an LDP-learned route to oban, despite the fact that none of the P routers are running LDP directly on their core facing interfaces. LDP-tunneling on the core RSVP LSPs allows the exchange of LDP labels over these LSPs. Figure 3.8 illustrates the label operations that occur on a packet following the lagavulin to oban path.

Notice the penultimate hop popping that occurs in Figure 3.8. Talisker's penultimate hop pops for the RSVP LSP from macduff to blair, sending just the LDP label to blair. Blair, as the penultimate hop for the LDP LSP from lagavulin to oban, pops the LDP

label, sending just the IP packet to oban.

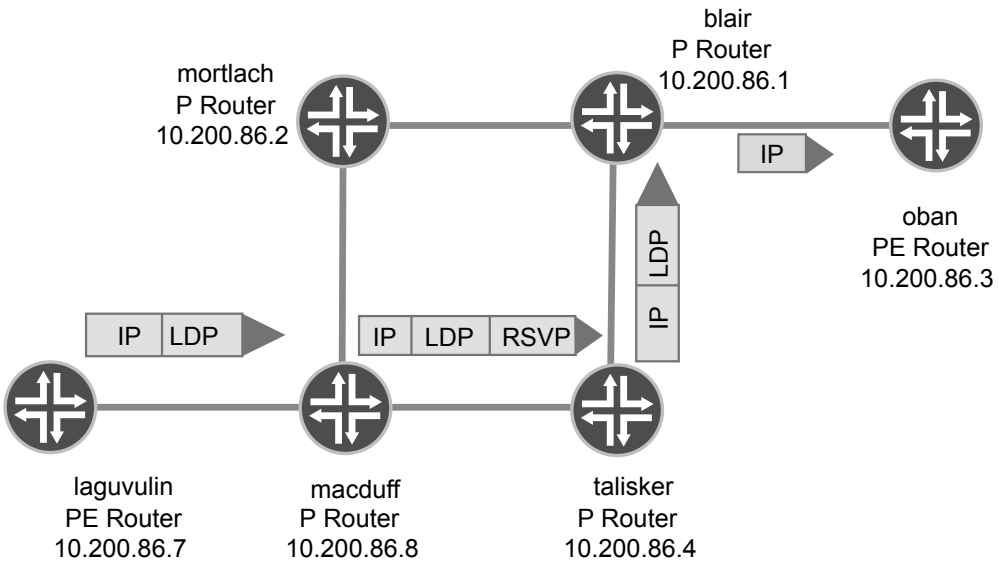



Figure 3.8 Label Operations with Tunneled-LDP

When the configuration of LDP tunneling is completed, the RSVP LSP's ingress and egress P routers form an LDP neighborship between their loopback addresses, (neighbors appear with lo0 as the neighbor interface in the output of show ldp neighbor command). The output shows the configuration required to enable ldp-tunneling:

```
ps@macduff> show configuration protocols mpls
label-switched-path macduff-to-blair {
   to 10.200.86.1;
   ldp-tunneling;
}
ps@macduff> show configuration protocols ldp
interface ge-0/0/2.0;
interface lo0.0;
```

The two critical pieces of configuration are the `ldp-tunneling` statement under the [protocols mpls label-switched-path] stanza, and secondly, because the LDP session establishes between the loopback interfaces, it's required that LDP be configured on the lo0.0 interface.

NOTE Both the macduff-to-blair *and* blair-to-macduff LSPs require `ldp-tunneling` to enable the bidirectional communication required for an LDP session. If the configuration is not made on both sides, a session will be shown in show ldp neighbor command with no label space ID and a hold time of 0 on the side with the configuration, as shown here:

```
ps@macduff> show ldp neighbor
Address          Interface        Label space ID       Hold time
192.168.86.2      ge-0/0/2.0       10.200.86.7:0          14
10.200.86.1       lo0.0           0.0.0.0:0               0
```

Following the configuration of the remote side, the new LDP session with the remote end of the LSP (blair in this case) should be properly shown in the output of `show ldp neighbor` command:

```
ps@macduff> show ldp neighbor
Address          Interface        Label space ID       Hold time
192.168.86.2      ge-0/0/2.0       10.200.86.7:0          12
10.200.86.1       lo0.0           10.200.86.1:0          43
```

Once you have configured the link between the PE and P routers with LDP, it should allow for edge-to-edge LDP label distribution. With this control-plane path established, the PE routers should have LDP labels for each other. Let's see:

```
ps@lagavulin> show route 10.200.86.3
inet.0: 16 destinations, 17 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.3/32     *[OSPF/10] 00:00:06, metric 4
                  > to 192.168.86.1 via ge-0/0/2.0
inet.3: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.3/32     *[LDP/9] 00:00:06, metric 1
                  > to 192.168.86.1 via ge-0/0/2.0, Push 104224
```

As expected, lagavulin has an inet.3 LDP route for oban's loopback address (10.200.86.3). In this configuration, only BGP is able to forward packets encapsulated within MPLS labels, because only BGP has access to the inet.3 table. To allow other protocols to have access to MPLS information, the `traffic-engineering bgp-igp` knob is required under the [protocols mpls] stanza:

```
ps@lagavulin> show configuration protocols mpls
traffic-engineering bgp-igp;
interface ge-0/0/2.0;
```

Adjusting this knob causes the router to move routes from the inet.3 routing table to inet.0 as shown here:

```
ps@lagavulin> show route 10.200.86.6
inet.0: 16 destinations, 20 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.6/32     *[LDP/9] 00:00:24, metric 1
                  > to 192.168.86.1 via ge-0/0/2.0, Push 104224
                  [OSPF/10] 00:00:24, metric 4

                  > to 192.168.86.1 via ge-0/0/2.0
```

NOTE Enabling the `traffic-engineering bgp-igp` knob causes the router to *remove the* route from the inet.3 table, which might break certain MPLS VPN implementations. To work around this problem, configure `traffic-engineering bgp-ibgp-both-ribs` under the [protocols mpls] stanza, which *copies the* route from inet.3 to inet.0, as shown here:

```
ps@lagavulin> show route 10.200.86.6
inet.0: 16 destinations, 20 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.6/32     *[LDP/9] 00:00:05, metric 1
                  > to 192.168.86.1 via ge-0/0/2.0, Push 104224
                  [OSPF/10] 00:14:57, metric 4
                  > to 192.168.86.1 via ge-0/0/2.0

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

```
10.200.86.6/32     *[LDP/9] 00:00:05, metric 1
                  > to 192.168.86.1 via ge-0/0/2.0, Push 104224
```

Lagavulin will attach a label of 104224 to packets destined to oban. When that label gets to macduff, macduff must perform a swap on the LDP label *and* a push for the RSVP-TE LSP:

```
ps@macduff> show route label 104224 detail
mpls.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
104224 (1 entry, 1 announced)
       *LDP   Preference: 9
              Next hop type: Router, Next hop index: 477
              Next-hop reference count: 2
              Next hop: 192.168.86.13 via ge-0/0/1.0 weight 0x1, selected
              Label-switched-path macduff-to-blair
              Label operation: Swap 101440, Push 100512(top)
              State: <Active Int>
              Age: 15:56     Metric: 1
              Task: LDP
              Announcement bits (1): 0-KRT
              AS path: I
              Prefixes bound to route: 10.200.86.6/32
```

After macduff pushes the RSVP label onto the stack and the packet forwarded, the next router in the path will only act on the topmost label, which is the RSVP label, meaning that the transit router, talisker in this case, need not know about the encapsulated LDP label and operates purely based on the RSVP label:

```
ps@talisker> show route label 100512 detail
mpls.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
100512 (1 entry, 1 announced)
       *RSVP  Preference: 7
              Next hop type: Router, Next hop index: 470
              Next-hop reference count: 3
              Next hop: 192.168.86.18 via fe-2/0/0.0 weight 0x1, selected
              Label-switched-path macduff-to-blair
              Label operation: Pop
              State: <Active Int>
              Age: 1:08:28   Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I
```

In this case, talisker's job is to pop the RSVP label (as it is the penultimate hop for the macduff-to-blair RSVP LSP), leaving just the LDP learned label when the packet is forwarded to blair. Blair will operate only on the topmost label (which is the LDP label learned over the tunneled LDP session):

```
ps@blair> show route label 101440 detail
mpls.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
101440 (1 entry, 1 announced)
       *LDP   Preference: 9
              Next hop type: Router, Next hop index: 473
              Next-hop reference count: 2
              Next hop: 192.168.86.25 via ge-6/0/0.0, selected
              Label operation: Pop
              State: <Active Int>
              Age: 32:25     Metric: 1
              Task: LDP
              Announcement bits (1): 0-KRT
              AS path: I
              Prefixes bound to route: 10.200.86.6/32
```

Blair will pop the LDP label, sending just the IP packet to oban, which would then perform a standard IP route lookup for forwarding. This hierarchical design uses label stacking to encapsulate an IP packet within an LDP label within an RSVP label for scalable edge-to-edge MPLS encapsulation. While the edge-to-core paths do not get the traffic engineering and resiliency advantages of RSVP signaled LSPs, bandwidth at the edge is usually cheap (when in the same facility) and LAN connections are invariably more stable than WAN connectivity, so these MPLS features are generally not as important on the edge.

## MPLS Scaling

MPLS provides lots of functionality that conventional IP methods cannot deliver. With this functionality comes great responsibility, as enabling MPLS places additional control load on the network. Additionally, some MPLS features such as VPLS expose the network to traffic types that it would not normally see. Both of these components require the network engineer to consider different scaling issues and design the network to limit the scaling factors or reduce their impact.

One of the major advantages of deploying MPLS in a network is the ability to traffic engineer LSPs. While it requires the network architecture to support the implementation, you will realize the benefits. Traffic engineering requires that the head-end of the LSP have a full picture of the network, both from an IGP standpoint as well as from a traffic-engineering database perspective. It necessitates a single OSPF area or single level IS-IS design. The increased processing performance and memory capacity of routers will dramatically expand the size limitations of a single-area or level design, so scaling is an important consideration to balance against these features when contemplating network evolution. It is also important that your IGP be used for only next-hop reachability and not carry network reachability, as this will allow your IGP TE domain to scale to a significantly larger size.

In addition to the LDP/RSVP hybrid design described earlier, there are other RSVP scaling mechanisms available. The two most commonly deployed RSVP scaling methods are hierarchical LSPs and a hybrid design similar to the LDP-tunneling method previously described, but using RSVP LSPs from the edge to the core, as well as within the core. Let's delve a little deeper into both of these methods of traffic engineering LSPs.

### Hierarchical LSPs

Hierarchical LSPs allow routers to consider core-to-core LSPs as *links*, which means the routers advertise their metrics and traffic-engineering characteristics (total bandwidth, available bandwidth, administrative-groups) via the IGP just as if they were real, physical circuits. The major benefit of this design is that the P routers have far fewer LSPs to keep track of. Figure 3.9 illustrates an example of this architecture.

NOTE OSPF is required when implementing hierarchical LSPs.

There are three steps in configuring hierarchical LSPs and all of them occur on the routers providing this LSP-in-LSP tunneling (the P routers, in our case). The first step is configuring an LSP as a te-link, which makes the *link* visible to other routers in the TED. The second step creates a *logical interface*. The OSPF and RSVP configurations use this logical interface in the same way they would a standard

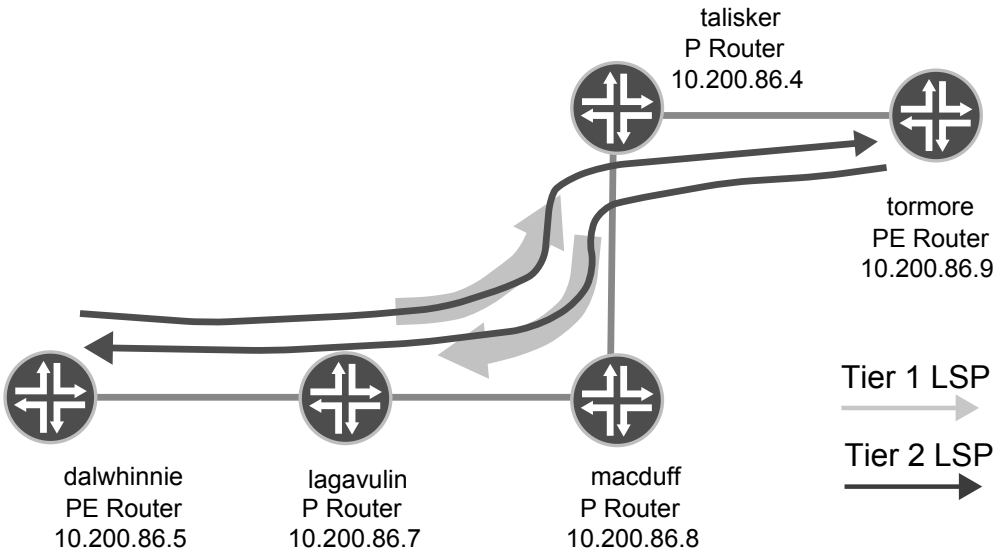



Figure 3.9 Hierarchical LSPs

physical interface. Notice that none of these steps involve any PE routers, which are blissfully ignorant to this tunneling that is happening in the core.

The following shows the configuration snippet for lagavulin, with the bolded items calling out pertinent configuration pieces. Following the configuration example, additional information provides explanations for the bolded items.

```
ps@lagavulin> show configuration protocols
rsvp {
    interface ge-0/0/1.0;
    interface ge-0/0/2.0;
    peer-interface peer-talisker;
}
mpls {
    label-switched-path lagavulin-to-talisker {
        to 10.200.86.4;
    }
    interface ge-0/0/1.0;
    interface ge-0/0/2.0;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/1.0
        interface ge-0/0/2.0;
        peer-interface peer-talisker;
    }
}
link-management {
    te-link lagavulin-to-talisker-te {
        local-address 192.168.87.1;
        remote-address 192.168.87.2;
        te-metric 1;
        label-switched-path lagavulin-to-talisker;
```

```
    }
    peer peer-talisker {
        address 10.200.86.4;
        te-link lagavulin-to-talisker-te;
    }
}
```

The peer-interface directives place the pseudo interfaces into the appropriate routing-protocols, namely OSPF and RSVP, but the link-management portion of the configuration is where things get interesting.

First, IP addresses provide layer-3 endpoints for the te-link, just like any other P2P interface or IP tunnel (192.168.87.1 and 192.168.87.2 in this case). A te-metric of this *link* is set to 1, which is akin to setting the OSPF metric of an interface and is the value used when comparing other possible paths. This means that the te-metric of a te-link must be better than the sum of the hops in the hierarchical LSPs path. Because the two-hop path from lagavulin-to-talisker has the default value of 2, a lower value (1) must be used to make it a more preferred path. The te-link is then bound to an LSP, lagavulin-to-talisker in this case. Finally, the peer interface is generated based on the te-link created in the previous step, and is the interface applied to the OSPF and RSVP configurations in the preceding lines. Note that the address of the peer is the same endpoint as the to directive in the label-switched-path configuration.

After committing, several commands will show the condition of the link-management tunnel. The most useful of these is the show link-management command as it aggregates all link-management information in one output.

TIP If there are too many link-management peers and te-links to show at one time, use the peer name peer name and te-link name te-link name arguments to limit the output.

Here the output shows lagavulin's link-management information:

```
ps@lagavulin> show link-management
Peer name: peer-talisker, System identifier: 55629
 State: Up, Control address: 10.200.86.4
 Hello interval: 150, Hello dead interval: 500
  TE links:
   lagavulin-to-talisker-te

 TE link name: lagavulin-to-talisker-te, State: Up
  Local identifier: 2684274792, Remote identifier: 2684274792,
  Local address: 192.168.87.1, Remote address: 192.168.87.2, Encoding: Packet,
  Switching: Packet, Minimum bandwidth: 0bps, Maximum bandwidth: 0bps,
  Total bandwidth: 0bps, Available bandwidth: 0bps
    Name                       State Local ID  Remote ID     Bandwidth Used  LSP-name
    lagavulin-to-talisker          Up      14956        0          0bps  Yes  dalwhinnie-to-tormore
```

The instrumentation shows you the te-link peer (10.200.86.4, talisker), the local and remote addresses for the tunnel (which other routers use to build their LSPs) and standard RSVP interface information such a maximum and available bandwidth. The container LSP is also shown, lagavulin-to-talisker in this case, and, potentially most important, any LSPs utilizing this hierarchical LSP. Here you can see that the dalwhinnie-to-tormore LSP is tunneling through this LSP.

Now that you know the LSP that is tunneling, you can use some other commands to verify that our hierarchical LSP is working as expected. The easiest method is to look at the path of the LSP. From dalwhinnie, let's use the show mpls lsp detail command to investigate its path:

```
ps@dalwhinnie> show mpls lsp name dalwhinnie-to-tormore detail
Ingress LSP: 1 sessions
10.200.86.9
  From: 10.200.86.5, State: Up, ActiveRoute: 0, LSPname: dalwhinnie-to-tormore
  ActivePath:  (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                 State: Up
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 60)
 192.168.86.29 S 192.168.87.2 S 192.168.86.33 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt):
         192.168.86.29 192.168.87.2 192.168.86.33
Total 1 displayed, Up 1, Down 0
```

Right off the bat, the output shows a very good sign: one of our tunnel interface addresses (192.168.87.2) shows up in the path, meaning that in all likelihood, this LSP is in fact tunneling through the hierarchical LSP. To verify the theory further, you can follow the label mappings across the path. As the edge-to-edge LSP runs from dalwhinnie to tormore, you can use tormore's endpoint address as a starting point on dalwhinnie, as such:

```
ps@dalwhinnie> show route 10.200.86.9 detail table inet.3

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
10.200.86.9/32 (1 entry, 0 announced)
       State: <FlashAll>
       *RSVP   Preference: 7
              Next-hop reference count: 3
              Next hop: 192.168.86.29 via ge-0/0/3.0 weight 0x1, selected
              Label-switched-path dalwhinnie-to-tormore
              Label operation: Push 300000
              State: <Active Int>
              Local AS:    1
              Age: 7:28      Metric: 61
              Task: RSVP
              AS path: I
```

You can specify the inet.3 table on dalwhinnie because neither protocols mpls traffic-engineering bgp-igp or bgp-igp-both-ribs are configured, so MPLS routes are stored only in the inet.3 table. Dalwhinnie is using RSVP label 300000 for this route and is forwarding the packet over fe-0/2/0, which is the connection to lagavulin. Moving us along to lagavulin, this configuration should provide additional interesting information. On lagavulin, you can look at the label swapping for label 300000 with a `show route` command:

```
ps@lagavulin> show route label 300000 detail
mpls.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
300000 (1 entry, 1 announced)
       *RSVP   Preference: 7/1
              Next hop type: Router, Next hop index: 526
              Next-hop reference count: 3
              Next hop: 192.168.86.1 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path lagavulin-to-talisker
              Label operation: Swap 299984, Push 299920(top)
              State: <Active Int AckRequest>
              Local AS:    1
              Age: 7:45      Metric: 1
              Task: RSVP
              Announcement bits (1): 0-KRT
              AS path: I
```

Under normal circumstances, lagavulin would simply swap the RSVP label and forward along the dalwhinnie-to-tormore LSP, but here you can actually see something different. Lagavulin is instead performing a swap and a push, finally forwarding the packet over the lagavulin-to-talisker hierarchical LSP. It looks somewhat similar to the LDP tunneling mechanism previously described, does it not?

In case you are not convinced that the LSP is really tunneling through another LSP, the final proof lies on a transit P in the hierarchical LSP's path. In this case, macduff proves the point:

```
ps@macduff> show mpls lsp
Ingress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Egress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 2 sessions
To            From          State   Rt Style Labelin Labelout LSPname
10.200.86.4    10.200.86.7    Up      1  1 FF  299920      3 lagavulin-to-talisker
10.200.86.7    10.200.86.4    Up      1  1 FF  299936      3 talisker-to-lagavulin
Total 2 displayed, Up 2, Down 0
```

If the LSP were not really tunneling, macduff would have to know about it, but it does not. It only knows about the core (hierarchical) LSPs from lagavulin to talisker and vice-versa, meaning that the dalwhinnie-to-tormore LSP is, in fact, tunneling through the lagavulin-to-talisker LSP without the edge PE or transit P routers knowing any better.

Hierarchical LSPs can be a powerful tool to manage a growing MPLS network with requirements that only RSVP-TE can fulfill.

## Hierarchical RSVP Domains

Core-to-edge RSVP LSP separation is also an option for scaling an RSVP domain. In its design shown in Figure 3.10, edge routers have RSVP-TE LSPs to their nearest core router(s) and the core routers form a full mesh. These *single-hop* LSPs separation enable scaling, as any given P router only needs to support as many LSPs as there are other P routers plus downstream PEs. The benefit of this design over the previously discussed LDP/RSVP hybrid architecture is that LSP protection mechanisms, such as link and link-node protections, can protect each hop in the path. It also allows for traffic engineering in the core as well as limited traffic engineering on the edge. It is important to note that core-to-edge RSVP LSP separation design *does not* support MPLS-based VPN services because there is no end-to-end MPLS path.

In Figure 3.10, the PE forwards traffic via an LSP to a nearby core P router. In this case, because this edge-to-core LSP is a single hop, the router does not push a label onto the stack (as the edge router is essentially the penultimate router). Once the packet gets to the P router, it forwards the packet via a core LSP to get it across the core, where the tail-end P router forwards the packet over a core-to-edge LSP.

NOTE The `protocols ospf|isis traffic-engineering shortcuts` and `protocols mpls traffic-engineering bgp-igp-both-ribs` are both required for this design to work.

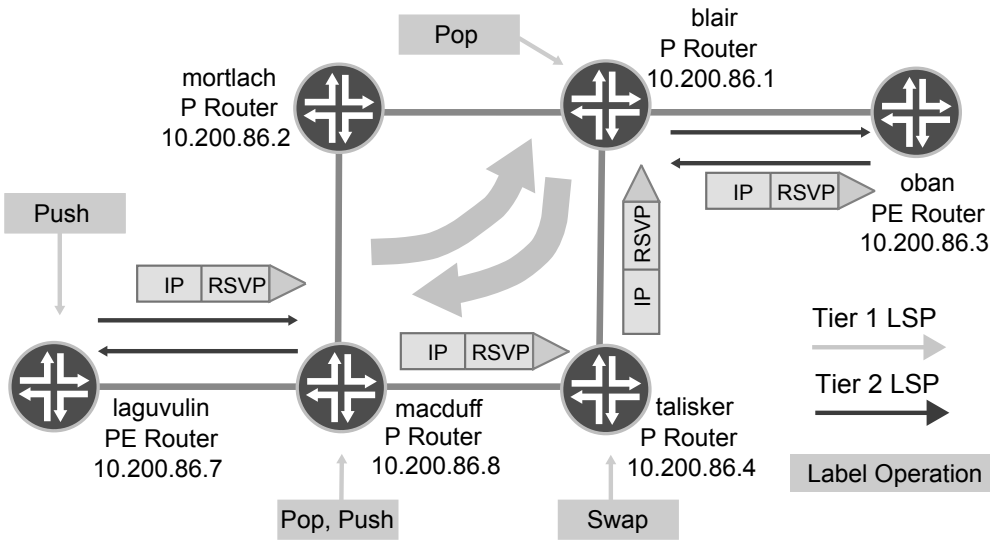



Figure 3.10 Edge-to-core and Core-to-core LSPs

Using the network shown in Figure 3.10, let's follow the flow of a packet entering on lagavulin and exiting from oban. The destination of this packet is 4.2.2.1 and oban is advertising 4.2.2.0/24 via BGP. Lagavulin will perform an IP route lookup and find that the next-hop for 4.2.2.0/24 is oban's loopback address, 10.200.86.3/32. Let's check:

```
ps@lagavulin> show route 4.2.2.1 detail
inet.0: 34 destinations, 45 routes (34 active, 0 holddown, 0 hidden)
4.2.2.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Next hop type: Indirect
              Next-hop reference count: 3
              Source: 10.200.86.3
              Next hop type: Router, Next hop index: 465
              Next hop: 192.168.86.1 via ge-0/0/2.0 weight 0x1, selected
              Label-switched-path lagavulin-to-macduff
              Protocol next hop: 10.200.86.3
              Indirect next hop: 8a6d000 131070
              State: <Active Int Ext>
              Local AS:    1 Peer AS:    1
              Age: 48:33     Metric2: 4
              Task: BGP_1.10.200.86.3+51109
              Announcement bits (2): 0-KRT 4-Resolve tree 2
              AS path: I
              Localpref: 100
              Router ID: 10.200.86.3
```

Lagavulin then performs a recursive lookup to this address and finds that it should forward it via the lagavulin-to-macduff PE-to-P LSP. Note that there is no label information shown in the following output because of the PHP behavior previously described:

```
ps@lagavulin> show route 10.200.86.3 detail
inet.0: 16 destinations, 18 routes (16 active, 0 holddown, 0 hidden)
10.200.86.3/32 (1 entry, 1 announced)
       *OSPF   Preference: 10
              Next hop type: Router, Next hop index: 462
```

```
            Next-hop reference count: 24
            Next hop: 192.168.86.1 via ge-0/0/2.0 weight 0x1, selected
            Label-switched-path lagavulin-to-macduff
            State: <Active Int>
            Age: 13:03     Metric: 4
            Area: 0.0.0.0
            Task: OSPFv2
            Announcement bits (1): 0-KRT
            AS path: I
```

Macduff receives the IP packet and performs the primary and recursive route lookups, and finds that it must forward this packet via a core LSP terminating on blair:

```
ps@macduff> show route 10.200.86.3 detail
inet.0: 18 destinations, 25 routes (18 active, 0 holddown, 0 hidden)
10.200.86.3/32 (1 entry, 1 announced)
       *OSPF   Preference: 10
            Next hop type: Router, Next hop index: 472
            Next-hop reference count: 10
            Next hop: 192.168.86.13 via ge-0/0/1.0 weight 0x1, selected
            Label-switched-path macduff-to-blair
            Label operation: Push 100544
            State: <Active Int>
            Age: 10:19     Metric: 3
            Area: 0.0.0.0
            Task: OSPFv2
            Announcement bits (1): 0-KRT
            AS path: I
```

The explicit route object (ERO) of this LSP shows the path the LSP is taking:

```
ps@macduff> show mpls lsp detail name macduff-to-blair | resolve
Ingress LSP: 5 sessions
blair
  From: macduff, State: Up, ActiveRoute: 0, LSPname: macduff-to-blair
  ActivePath:  (primary)
  LoadBalance: Random
  Autobandwidth
  AdjustTimer: 3600 secs
  Max AvgBW util: 0bps, Bandwidth Adjustment in 886 second(s).
  Overflow limit: 0, Overflow sample count: 0
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                 State: Up
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 20)
 talisker S blair S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt):
         192.168.86.13 192.168.86.18
Total 1 displayed, Up 1, Down 0
Egress LSP: 5 sessions
Total 0 displayed, Up 0, Down 0
Transit LSP: 4 sessions
Total 0 displayed, Up 0, Down 0
```

And it happens to traverse talisker, so let's look there for the label operation for 100544:

```
ps@talisker> show route label 100544 detail
mpls.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
100544 (1 entry, 1 announced)
       *RSVP   Preference: 7
            Next hop type: Router, Next hop index: 464
            Next-hop reference count: 3
```

```
            Next hop: 192.168.86.18 via fe-2/0/0.0 weight 0x1, selected
            Label-switched-path macduff-to-blair
            Label operation: Pop
            State: <Active Int>
            Age: 17:58     Metric: 1
            Task: RSVP
            Announcement bits (1): 0-KRT
            AS path: I
```

Unsurprisingly, talisker, as the penultimate hop, will penultimate hop pop (PHP) the packet's label, forwarding just the IP packet to blair. The route lookup process repeats on blair, instructing it to forward the packet on the blair-to-oban, P to PE LSP:

```
ps@blair> show route 10.200.86.3 detail
inet.0: 18 destinations, 25 routes (18 active, 0 holddown, 0 hidden)
10.200.86.3/32 (2 entries, 1 announced)
      State: <FlashAll>
      *RSVP  Preference: 7
            Next hop type: Router, Next hop index: 475
            Next-hop reference count: 6
            Next hop: 192.168.86.25 via ge-6/0/0.0 weight 0x1, selected
            Label-switched-path blair-to-oban
            State: <Active Int>
            Age: 12:18     Metric: 1
            Task: RSVP
            Announcement bits (2): 0-KRT 2-OSPFv2
            AS path: I
```

Because of the chain of LSPs, lagavulin was able to forward a packet to oban via RSVP signaled LSPs without having an end-to-end LSP. While the design is popular, it does require all routers, P and PE alike, to maintain a full routing table as P routers will have to perform IP route lookups as the tails of PE-to-P and P-to-P LSPs. Figure 3.10 illustrates this concept.

## RSVP Refresh Reduction

RSVP, by default, uses soft, periodic messaging to maintain reservations and exchange control messages between neighbors. The soft messaging can lead to scalability issues as well as reliability and latency problems. The scalability issues arise from the sending and processing of the RSVP refresh message, and the scaling problem arises with the number of RSVP sessions. Missed one-time messages, such as PathTear, or PathErr messages, or slow failure detection, which can be the result of missed periodic refresh timers, can also affect traffic. In other words, there can easily be a delay in detection because there is a built-in delay in messaging. To improve the scalability and reliability of these messages, RFC 2961 describes three new extensions to RSVP: reliable messages, bundle messages, and summary refresh messages.

- Reliable messages: These messages add reliability to RSVP message exchanges by introducing a MESSAGE_ID object to RSVP messages. If the ACK_Desired field is set on RSVP messages, the receiver is to respond with a MESSAGE_ID_ACK message, which includes the MESSAGE_ID, allowing the receiver to confirm receipt of the message.
- Bundle messages: A bundle message groups several standard RSVP messages into a single message. These can include any of the standard RSVP messages

including Path, PathErr, Resv, ResvTear, ResvErr, ResvConf, and ACK messages.

- Summary refresh messages: Summary refresh messages aggregate several Path and Resv messages into a single update. MESSAGE_IDs are mapped to Path or Resv state, and a MESSAGE_ID existing in a summary refresh message serves to update the associated Path or Resv state.

Testing found that approximately 7500 LSPs required approximately 235 RSVP messages per second. The output below shows that in 119 seconds, oban received nearly 28000 RSVP messages.

```
ps@oban> show rsvp session | count
Count: 7442 lines
ps@oban> show system uptime | match current
Current time: 2010-12-09 22:18:30 UTC
ps@oban> show rsvp statistics | match "^  Path  |Sent.*Received"
Sent     Received       Sent     Received
Path              0          1         0          0
ps@oban> show system uptime | match current
Current time: 2010-12-09 22:20:29 UTC
ps@oban> show rsvp statistics | match "^  Path  |Sent.*Received"
Sent     Received       Sent     Received
Path              3       27978         0        1269
```

That is a lot of work for the routing-engine to be doing when there is an alternative to reduce this load. Configuring RSVP refresh reduction is very simple. All that is necessary is the addition of the `aggregate` statement, applied to the interface(s) under the [protocols rsvp] hierarchy. For example, on oban, ge-0/0/2.0 includes this configuration:

```
ps@oban> show configuration protocols rsvp
   interface ge-0/0/2.0 {
   aggregate;
}
```

After enabling RSVP refresh-reduction on our RSVP interfaces, our packet per-second rate drops to approximately 12. RSVP refresh reduction has reduced the RSVP packets the routing-engines need to transmit and process by 95%!

```
ps@oban> show system uptime | match current
Current time: 2010-12-09 22:40:31 UTC

ps@oban> show rsvp statistics | match "^  Path  |Sent.*Received"
               Sent     Received       Sent     Received
  Path              0          0          0          0

ps@oban> show system uptime | match current
Current time: 2010-12-09 22:42:29 UTC

ps@oban> show rsvp statistics | match "^  Path  |Sent.*Received"
               Sent     Received       Sent     Received
  Path              3        1560          0          4
```

In short, the reduced messaging decreases the overhead on the network and the addition of the MESSAGE_ID and MESSAGE_ID_ACK objects provides reliable messages, allowing for improved fault detection and network convergence.

## BGP Considerations

Another important consideration in an MPLS network is the impact of MPLS on a BGP deployment, especially in environments that utilize route-reflection. In this section, VPN next-hop resolution on route-reflectors and the BGP route-target family are both detailed.

### Route-reflection in VPN Environments

When scaling BGP in an MPLS VPN environment, MPLS next-hop reachability is a important trait for consideration in your network. When using route-reflection as a BGP scaling mechanism, the route-reflector must have MPLS label information for any PE routers for which it is reflecting MPLS VPN routes. If the route-reflector does not have an MPLS label for the PE, it will invalidate VPN routes with a next-hop type of *Unusable*.

There are two options to provide the route-reflector with MPLS label information for the PE routers, and the choice depends on which label distribution mechanism is used. In an LDP environment, simply allowing the route-reflector to participate in LDP provides it with the inet.3 routes necessary to resolve VPN route next-hops. Providing inet.3 routing information in RSVP environments is a bit more complicated because an RSVP only environment requires LSPs from each PE to the route-reflector(s). Let's look at Figure 3.11 for a moment.

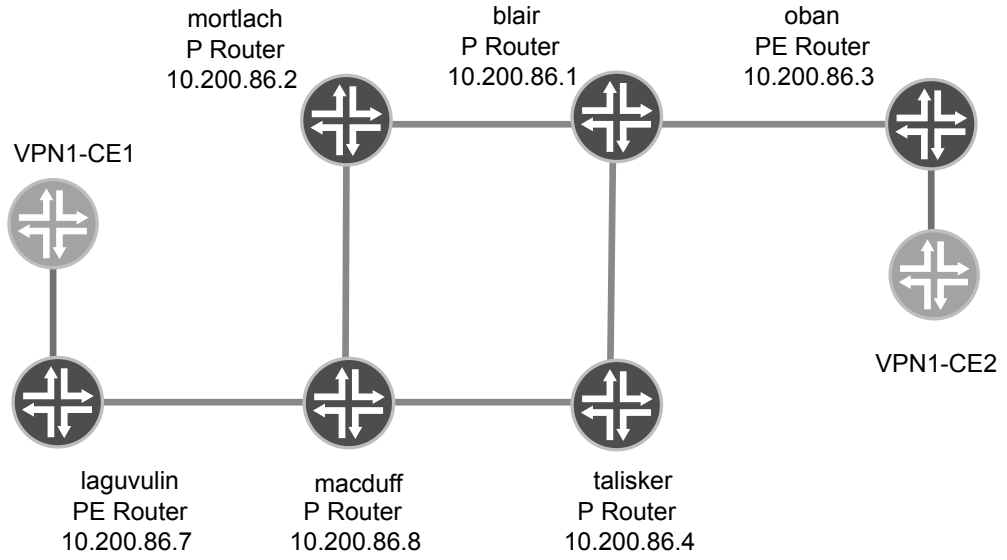



Figure 3.11 MPLS VPN with Route-reflection

In this network, the two PE routers, lagavulin and oban, support a single CE in the same 4364-style VPN called *VPN-1*. These PE routers share BGP routes via a route-reflector, which is served by blair. Without MPLS to the route-reflector, it will invalidate any VPN routes and therefore, not advertise these routes to the other PE.

The `show route` output displays that the loopback addresses of the PE routers are only known via OSPF (and not LDP or RSVP):

```
ps@blair> show route 10.200.86.3
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.3/32     *[OSPF/10] 01:44:13, metric 1
                  > to 192.168.86.25 via 6/0/0.0

ps@blair> show route 10.200.86.7
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.7/32     *[OSPF/10] 01:42:44, metric 3
                    to 192.168.86.49 via ge-0/0/2.0
                  > to 192.168.86.17 via ge-0/0/3.0
```

Let's show an example of what the hidden routes would look like on the route-reflector in the *absence* of MPLS. As you can see, the next hop type is `Unusable`:

```
ps@blair> show route protocol bgp hidden detail
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
__juniper_private2__.inet.0: 1 destinations, 1 routes (0 active, 0 holddown, 1 hidden)
bgp.l3vpn.0: 2 destinations, 2 routes (0 active, 0 holddown, 2 hidden)
1:1:1.1.1.0/24 (1 entry, 0 announced)
        BGP    Preference: 170/-101
               Route Distinguisher: 1:1
               Next hop type: Unusable
               Next-hop reference count: 2
               State: <Hidden Int Ext>
               Local AS:     1 Peer AS:     1
               Age: 1:18:09
               Task: BGP_1.10.200.86.7+179
               AS path: I
               Communities: target:1:1
               VPN Label: 16
               Localpref: 100
               Router ID: 10.200.86.7
1:2:2.2.2.0/24 (1 entry, 0 announced)
        BGP    Preference: 170/-101
               Route Distinguisher: 1:2
               Next hop type: Unusable
               Next-hop reference count: 2
               State: <Hidden Int Ext>
               Local AS:     1 Peer AS:     1
               Age: 1:18:13
               Task: BGP_1.10.200.86.6+179
               AS path: I
               Communities: target:1:1
               VPN Label: 16
               Localpref: 100
               Router ID: 10.200.86.3
```

And because this path is unusable, the route-reflector will not advertise these routes to the PE routers:

```
ps@blair> show route advertising-protocol bgp 10.200.86.3
ps@blair> show route advertising-protocol bgp 10.200.86.7
ps@blair>
```

But enabling LDP across the network resolves the problem, as the route-reflector will have an MPLS path to the PE. After turning LDP on, neighboring routers advertise labels for the PE loopbacks (for clarity, the bold lines show the LDP routes):

```
ps@blair> show route 10.200.86.3
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
```

```
+ = Active Route, - = Last Active, * = Both
10.200.86.3/32     *[OSPF/10] 01:46:16, metric 1
                  > to 10.100.7.3 via ge-6/0/0.0
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.3/32     *[LDP/9] 00:01:15, metric 1
                  > to 10.100.7.3 via ge-6/0/0.0
ps@blair> show route 10.200.86.7
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.86.7/32     *[OSPF/10] 01:46:18, metric 3
                    to 192.168.86.49 via ge-0/0/2.0
                  > to 192.168.86.17 via ge-0/0/3.0
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.7/32     *[LDP/9] 00:01:17, metric 1
                  > to 192.168.86.49 via ge-0/0/2.0, Push 103168
                    to 192.168.86.17 via ge-0/0/3.0, Push 100592
```

Due to the new MPLS paths to the advertising PE router, the route-reflector marks the VPN routes as active and begins advertising them to the opposite PE router, as shown here:

```
ps@blair> show route protocol bgp detail
inet.0: 18 destinations, 21 routes (18 active, 0 holddown, 0 hidden)
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
__juniper_private2__.inet.0: 1 destinations, 1 routes (0 active, 0 holddown, 1 hidden)
mpls.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
1:1:1.1.1.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Route Distinguisher: 1:1
              Next hop type: Indirect
              Next-hop reference count: 1
              Source: 10.200.86.7
              Protocol next hop: 10.200.86.7
              Push 16
              Indirect next hop: 2 no-forward
              State: <Active Int Ext>
              Local AS:    1 Peer AS:    1
              Age: 12:25     Metric2: 1
              Task: BGP_1.10.200.86.7+56251
              Announcement bits (1): 0-BGP RT Background
              AS path: I
              Communities: target:1:1
              VPN Label: 16
              Localpref: 100
              Router ID: 10.200.86.7

1:2:2.2.2.0/24 (1 entry, 1 announced)
       *BGP    Preference: 170/-101
              Route Distinguisher: 1:2
              Next hop type: Indirect
              Next-hop reference count: 1
              Source: 10.200.86.3
              Protocol next hop: 10.200.86.3
              Push 16
              Indirect next hop: 2 no-forward
              State: <Active Int Ext>
              Local AS:    1 Peer AS:    1
              Age: 1:32:20    Metric2: 1
```

```
            Task: BGP_1.10.200.86.6+179
            Announcement bits (1): 0-BGP RT Background
            AS path: I
            Communities: target:1:1
            VPN Label: 16
            Localpref: 100
            Router ID: 10.200.86.3

ps@blair> show route advertising-protocol bgp 10.200.86.7
bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix              Nexthop           MED    Lclpref    AS path
  1:1:1.1.1.0/24
*                   10.200.86.7              100        I
ps@blair> show route advertising-protocol bgp 10.200.86.3
bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix              Nexthop           MED    Lclpref    AS path
  1:2:2.2.2.0/24
*                   10.200.86.3              100        I
```

Now, obviously this situation is different in an RSVP-only network. An RSVP network requires that there be LSPs from the route-reflector to the PE routers, unless the operator implements the steps described below.

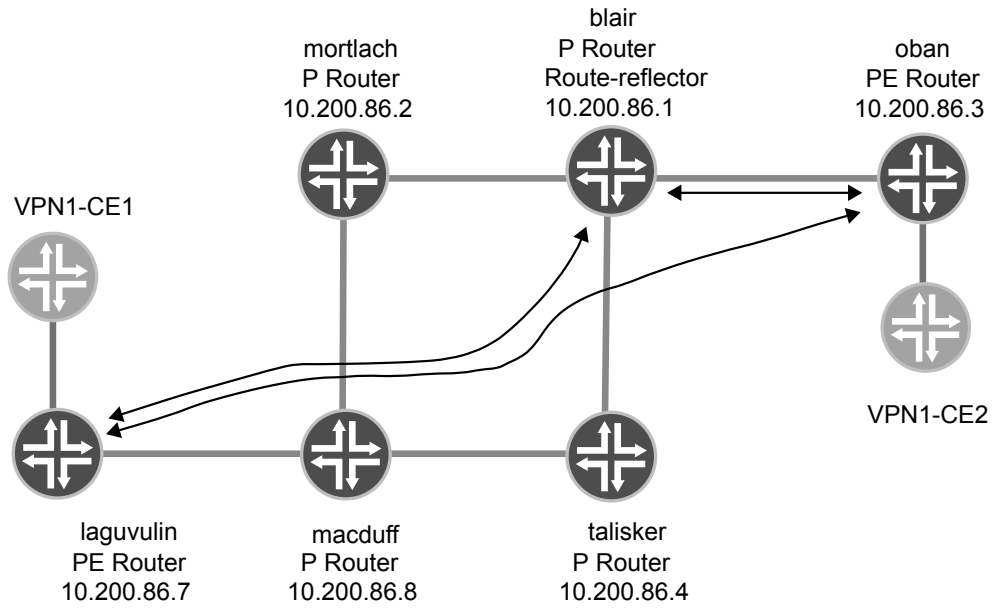



Figure 3.12 MPLS VPN with Route-reflection and RSVP

The important aspect of Figure 3.12 is that the route-reflector must have (or at least think it has, as you will see later) an MPLS path to the next-hop of the advertising PE router. LDP is certainly the simpler model, but it may not be consistent with the MPLS design of the network.

## Allowing VPN next-hop Resolution on Route-reflectors

NOTE This recommendation is specific to environments in which the route-reflector is either *not* in the forwarding plane, or is a pure P router.

As mentioned previously, the route-reflector has to *think* it has an MPLS path to PE routers to validate VPN routes. While that section described a method in which the route-reflector actually has MPLS paths to the PE routers, this section describes ways to make the route-reflector think it does. There are two methods to achieve this behavior, the first is using the Junos resolution keyword under the [routing-options] hierarchy and the second is a more complicated design utilizing rib-groups.

The resolution feature is the easiest way to allow a route-reflector that does not have MPLS reachability to a PE to validate next-hop reachability, allowing the advertisement of VPN routes to other PEs. As shown before, without next-hop resolution, VPN routes show as *Unusable*, which prevents the installation or advertisement of the routes. Allowing the bgp.l3vpn.0 table to use another routing-table for resolution is a quick way of solving this problem. Before enabling, let's check that the route-reflector is invalidating the VPN routes due to the unusable path (key information shown in bold):

```
ps@blair> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table         Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0               0          0          0          0          0          0
bgp.l3vpn.0          2          0          0          0          0          0
Peer             AS     InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Damped...
10.200.86.3        1        82         83       0       2       35:18 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 0/1/0
10.200.86.7        1        80         81       0      10       34:36 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 0/1/0
```

The configuration here instructs the router to use inet.0 to resolve routes in the bgp.l3vpn.0 routing table:

```
ps@blair> show configuration routing-options
router-id 10.200.86.1;
autonomous-system 1;
resolution {
    rib bgp.l3vpn.0 {
        resolution-ribs inet.0;
    }
}
```

After committing, the route-reflector can validate the routes and begins advertising them to other PEs (again, the pertinent information is bolded for clarity):

```
ps@blair> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table         Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0               0          0          0          0          0          0
bgp.l3vpn.0          2          2          0          0          0          0
Peer             AS     InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Damped...
10.200.86.3        1        86         87       0       2       36:54 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
10.200.86.7        1        84         85       0      10       36:12 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
```

That was the simple approach. It is also possible to use a more complex design utilizing rib-groups to share inet.0 routing information with inet.3, allowing VPN route validation. However, before getting into the specifics, rib-groups in general

need further explanation, as they are one of the more confusing and frequently misunderstood concepts in the Junos operating system.

Rib-groups are a collection of routing-tables that, among other things, allow routing-protocols to install routes in more than one routing table. There are many other applications for rib-groups, but for the purposes of this discussion and rib-group clarity, let's focus on one particular use case. Say you want to install OSPF routes into both inet.0 and inet.3. There are two steps to this process, creating the rib-group and then directing the routing protocol to install routes into the newly created rib-group.

NOTE Creating the rib-group requires a rib-group name, which can be any name you wish, but descriptive names provide increased operational value.

The first routing-table listed is the primary table, or the default table for the instance of the protocol. As this OSPF instance is running in the main (inet.0) routing instance, inet.0 is the primary table. In a rib-group, the configuration includes the primary routing-table first, followed by any subsequent tables, making the rib-groups configuration look like the following:

```
ps@blair> show configuration routing-options
rib-groups {
    inet.0-to-inet.3 {
        import-rib [ inet.0 inet.3 ];
    }
}
```

OSPF has to know to install routes into the inet.0-to-inet.3 rib-group, rather than just the inet.0 table, which is default OSPF behavior. To do this, use the `rib-group` statement under the [protocols ospf] hierarchy:

```
ps@blair> show configuration protocols ospf
rib-group inet.0-to-inet.3;
area 0.0.0.0 {
    interface ge-0/0/1.0 {
        interface-type p2p;
    }
    interface ge-0/0/2.0 {
        interface-type p2p;
    }
    interface ge-0/0/3.0 {
        interface-type p2p;
    }
    interface ge-6/0/0.0 {
        interface-type p2p;
    }
    interface lo0.0;
}
```

Following this change, inet.0 OSPF routes now exist in the inet.3 table (the output below only shows part of the full routing-table for brevity):

```
ps@blair> show route table inet.3
inet.3: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.5/32     *[OSPF/10] 00:03:29, metric 1
                  > to 192.168.86.10 via ge-0/0/1.0
10.200.86.6/32     *[OSPF/10] 00:03:29, metric 1
                  > to 192.168.86.10 via ge-0/0/1.0
192.168.86.12/30     *[OSPF/10] 00:03:29, metric 3
                  > to 192.168.86.49 via ge-0/0/2.0
```

```
                to 192.168.86.17 via ge-0/0/3.0
[…]
```

Because the PE loopbacks are in the inet.3 table, the route-reflector can resolve the next-hops for the VPN routes and can install and advertise them to other PEs as shown:

```
ps@blair> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table         Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0               0          0          0          0          0          0
bgp.l3vpn.0          2          2          0          0          0          0
Peer             AS     InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Damped...
10.200.86.3       1       225        236       0       2     1:38:53 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
10.200.86.7       1       225        234       0      10     1:38:11 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
```

One unnecessary consequence to this configuration is that it copies all OSPF routes into inet.3, and you can see in the summary output, that in addition to the /32 loopback routes, /30 connection networks are also present. The inet.3 table does not need these routes, in fact, only the /32 loopback routes are needed.

Junos has two solutions to this problem, one is a simple default route in the inet.3 table and the second is a more complex solution, which controls route copying through policy.

In Junos, the user can configure static routes within individual rib tables. This is done at the [routing-options] hierarchy. The following configuration provides an example of such a configuration:

```
routing-options {
   rib inet.3 {
      static {
         route 0.0.0.0/0 discard;
      }
   }
}
```

For additional control, policies can limit route copying into the inet.3 table. These policies look like any other routing policy, but are applied to the rib-group via the `import-policy` statement. In the example network, all loopbacks received addresses out of the 10.200.86.0/24 subnet, which makes controlling route copying through policy much simpler. The following configuration snippet provides an example of this approach:

```
ps@blair> show configuration policy-options policy-statement Loopbacks-Only
term Loopbacks {
   from {
      route-filter 10.200.86.0/24 prefix-length-range /32-/32;
   }
   then accept;
}
term Reject-All-Else {
   then reject;
}
```

And the policy only matches /32 routes out of the 10.200.86/24 subnet as shown:

```
ps@blair> show configuration routing-options
rib-groups {
   inet.0-to-inet.3 {
      import-rib [ inet.0 inet.3 ];
      import-policy Loopbacks-Only;
   }
}
```

Now that the router is filtering the routes it copies from inet.0 to inet.3, inet.3 only contains the PE loopback addresses:

```
ps@blair> show route table inet.3
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.200.86.2/32     *[OSPF/10] 00:00:58, metric 1
                  > to 10.100.7.17 via ge-0/0/1.402
10.200.86.3/32     *[OSPF/10] 00:00:58, metric 1
                  > to 10.100.7.6 via ge-0/0/1.405
10.200.86.4/32     *[OSPF/10] 00:00:58, metric 1
                  > to 10.100.7.10 via ge-0/0/1.404
10.200.86.7/32     *[OSPF/10] 00:00:58, metric 3
                    to 10.100.7.17 via ge-0/0/1.402
                  > to 10.100.7.10 via ge-0/0/1.404
10.200.86.8/32     *[OSPF/10] 00:00:58, metric 2
                    to 10.100.7.17 via ge-0/0/1.402
                  > to 10.100.7.10 via ge-0/0/1.404
```

As before, the process allows the PE to install and advertise the VPN routes as shown here:

```
ps@blair> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table         Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0                0          0          0          0          0          0
bgp.l3vpn.0           2          2          0          0          0          0
Peer             AS     InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Damped...
10.200.86.3       1       243       256       0       2    1:47:11 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
10.200.86.7       1       244       255       0      10    1:46:29 Establ
  inet.0: 0/0/0
  bgp.l3vpn.0: 1/1/0
```

These two methods accomplish the same basic goal, allowing a route-reflector to resolve VPN next-hops without MPLS information for the PE loopbacks. Setting the `routing-options resolution` Junos keywords is clearly the more simplistic approach, while rib-groups offer more control and the expense of complexity. In most environments, routing-options resolution is the appropriate method.

## BGP Route-target Family

Before leaving the topic of VPNs, another scaling concern is the number of routes advertised between PEs (and to a lesser degree, between PEs and route-reflectors). The default behavior of PE routers in a VPN is to advertise all VPN routes to peers configured with family inet-vpn, and it is up to the receiving PE to decide which to keep and which to ignore based on the import policies of its local VPNs.

But Junos has an answer to this scaling concern in the form of a new BGP family called *route-target*. Networks that do not use route-reflection most commonly use this feature, and if a route-reflector is used, it most likely needs to receive all VPN routes to guarantee appropriate distribution.

If it is desirable to use the route-target family in your route-reflection environment, utilize the Junos `advertise-default` knob. The `advertise-default route-target` Junos keywords advertise the default (0:0:0/0) route-target, suppressing all specifics. This allows a route-reflector to receive routes from all route targets without having to specify each route-target individually. To more accurately illustrate this concept, Figure 3.13 updates our network to remove the route-reflector, requiring the PE routers to peer directly. Additionally two new PE routers grow the network to better show the functionality of the route-target feature.

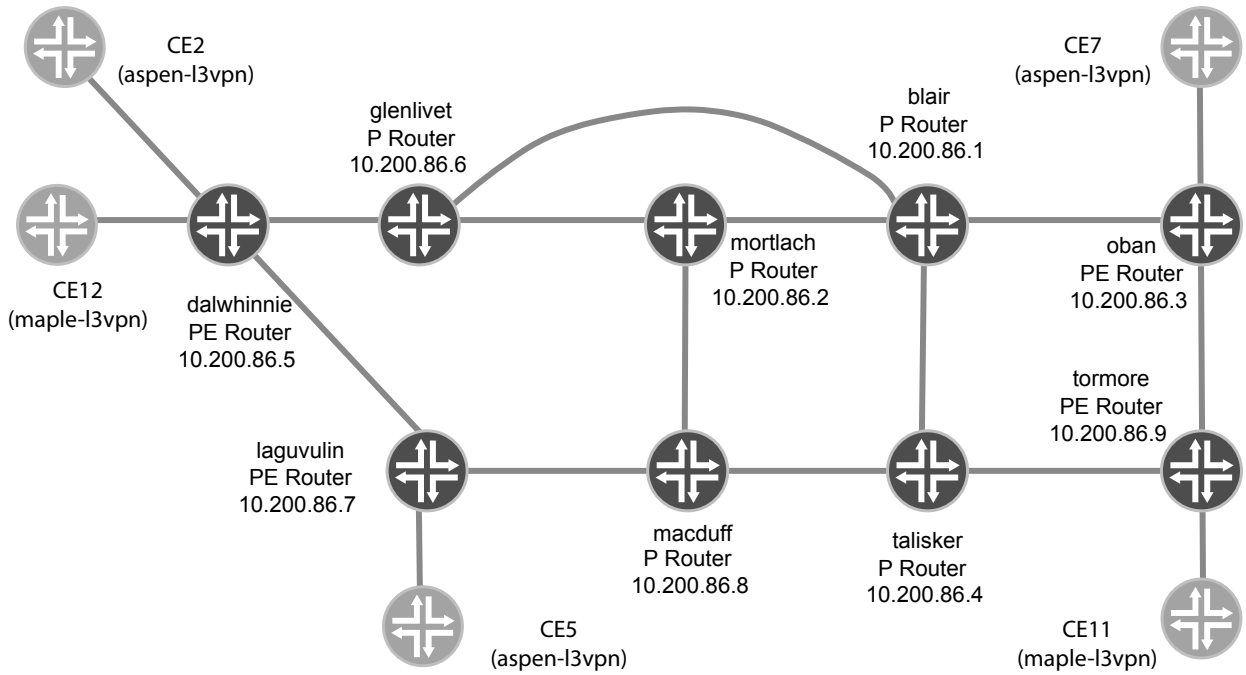



Figure 3.13 Route-target Base Topology

The topology of Figure 3.13 includes two VPNs, one called *aspen* that resides on dalwhinnie, lagavulin, and oban, and *maple,* which exists on dalwhinnie and tormore. Based on this design, oban and lagavulin only need aspen routes, tormore only needs maple routes and dalwhinnie needs both sets. Under default circumstances, the four PE routers advertise all VPN routes to all of the other PE routers, whether or not the remote PE has a VPN importing routes with a given target. Using tormore as an example, the only remote PE that is interested in tormore's maple routes is dalwhinnie, but tormore is advertising maple routes to all three PEs as shown here:

```
ps@tormore> show bgp summary
Groups: 1 Peers: 3 Down peers: 0
Table         Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0           1          1          0          0          0          0
Peer              AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Rece
ived/Damped...
10.200.86.3        1         19         20       0       0        7:36 Establ
  bgp.l3vpn.0: 0/0/0
10.200.86.5        1         18         20       0       0        7:30 Establ
  bgp.l3vpn.0: 1/1/0
  maple.inet.0: 1/1/0
10.200.86.7        1         20         21       0       0        7:42 Establ
```

```
  bgp.l3vpn.0: 0/0/0

ps@tormore> show route advertising-protocol bgp 10.200.86.3    ß oban
maple.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
* 5.5.5.0/24            Self                    100       I

ps@tormore> show route advertising-protocol bgp 10.200.86.5     ß dalwhinnie
maple.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
* 5.5.5.0/24            Self                    100       I

ps@tormore> show route advertising-protocol bgp 10.200.86.7     ß lagavulin
maple.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
  Prefix               Nexthop           MED    Lclpref    AS path
* 5.5.5.0/24            Self                    100       I
```

As you can see, oban and lagavulin are receiving maple's 5.5.5.0/24 route even though they will never use it. This is where the route-target family comes in. The Junos route-target BGP family allows a PE router to let other PE routers know which VPN route-targets it will import, meaning remote PE routers can limit VPN route-advertisements to only those that the receiving PE use. The configuration is simple and requires only the addition of the `route-target` statement to the appropriate BGP groups, here the *ibgp* bgp group (advertised route-target routes are stored in the bgp.rtarget.0 route-table):

```
ps@tormore> show configuration protocols bgp
group ibgp {
    type internal;
    local-address 10.200.86.9;
    family inet-vpn {
        unicast;
    }
    family route-target;
    neighbor 10.200.86.7;
    neighbor 10.200.86.3;
    neighbor 10.200.86.5;
}
```

After configuration, use the `show route table bgp.rtarget.0` command to display learned route-targets, like this example from tormore:

```
ps@tormore> show route table bgp.rtarget.0
bgp.rtarget.0: 2 destinations, 5 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1:1:1/96
                *[BGP/170] 00:01:58, localpref 100, from 10.200.86.3
                  AS path: I
                 > to 192.168.86.38 via ge-0/0/3.0
                 [BGP/170] 00:01:54, localpref 100, from 10.200.86.5
                  AS path: I
                 > to 192.168.86.38 via ge-0/0/3.0, Push 100496
                   to 192.168.86.34 via ge-0/0/2.0, Push 655505
                 [BGP/170] 00:02:03, localpref 100, from 10.200.86.7
                  AS path: I
                 > to 192.168.86.34 via ge-0/0/2.0, Push 655489
1:2:2/96
                *[RTarget/5] 00:02:15
                  Local
                 [BGP/170] 00:01:54, localpref 100, from 10.200.86.5
                  AS path: I
                 > to 192.168.86.38 via ge-0/0/3.0, Push 100496
                   to 192.168.86.34 via ge-0/0/2.0, Push 655505
```

Careful examination of the output shows a few key items. First, it tells us that 10.200.86.3 (oban), 10.200.86.5 (dalwhinnie), and 10.200.86.7 (lagavulin) all have VRFs importing routes with the 1:1 route-target, and secondly, that 10.200.86.5 (dalwhinnie) will use routes with the 2:2 target. Local indicates that the local router (tormore) also imports routes with the 2:2 target.

Based on this information, the PE routers can filter VPN route advertisement to remote PEs based on which route-targets those PEs have advertised. Now, with the Junos route-target family enabled, tormore more intelligently advertises VPN routes, meaning it is only advertising maple routes to dalwhinnie, as shown here:

```
ps@tormore> show route advertising-protocol bgp 10.200.86.3             ß oban
bgp.rtarget.0: 2 destinations, 5 routes (2 active, 0 holddown, 0 hidden)
Prefix               Nexthop           MED    Lclpref    AS path
1:2:2/96
*                     Self                    100       I
ps@tormore> show route advertising-protocol bgp 10.200.86.5     ß dalwhinnie
maple.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
Prefix               Nexthop           MED    Lclpref    AS path
* 5.5.5.0/24           Self                    100       I
bgp.rtarget.0: 2 destinations, 5 routes (2 active, 0 holddown, 0 hidden)
Prefix               Nexthop           MED    Lclpref    AS path
1:2:2/96
*                     Self                    100       I
ps@tormore> show route advertising-protocol bgp 10.200.86.7     ß lagavulin
bgp.rtarget.0: 2 destinations, 5 routes (2 active, 0 holddown, 0 hidden)
Prefix               Nexthop           MED    Lclpref    AS path
1:2:2/96
*                     Self                    100       I
```

As you can see, the procedure could greatly reduce the network overhead in any large VPN deployments.

## VPLS

VPLS offers significant feature and flexibility improvements over other Layer 2 VPN technologies, but also has some scalability concerns. The most pressing of these concerns is how to manage broadcast and unknown unicast and multicast traffic (affectionately referred to as *BUM traffic*). It is of particular concern in VPLS environments because these types of traffic are both commonly experienced and are flooded to all remote sites.

Point-to-multipoint (P2MP) LSPs efficiently flood BUM traffic, reducing the network load and firewall filters to allow engineers to limit the transmission of BUM traffic over a VPLS domain. Let's explore.

### P2MP LSPs

Most of the LSPs discussed thus far have been point-to-point (P2P) LSPs. These types of LSPs can be, and in fact are, used in VPLS forwarding, but forwarding BUM traffic over P2P LSPs is inefficient because the path to different LSPs often follows a similar path across the core, and using P2P LSPs for this traffic duplicates BUM traffic over these paths.

P2MP LSPs solve the problem by implementing an LSP that is rooted at the source PE, but branches out to all receiver PEs within a VPLS instance, allowing common links in different paths to carry only a single copy of the packet. The packet replicates only at branch points. It's very similar to IP multicast forwarding and it's

important to understand that P2MP LSPs only carry BUM traffic; P2P LSPs carry traffic destined for known MAC addresses and this is unicast traffic. Figures 3.14 and 3.15 illustrate the difference between P2P and P2MP forwarding. The first shows the packet flows for BUM traffic in a P2P environment, while the second shows the same flows with P2MP LSPs in use.

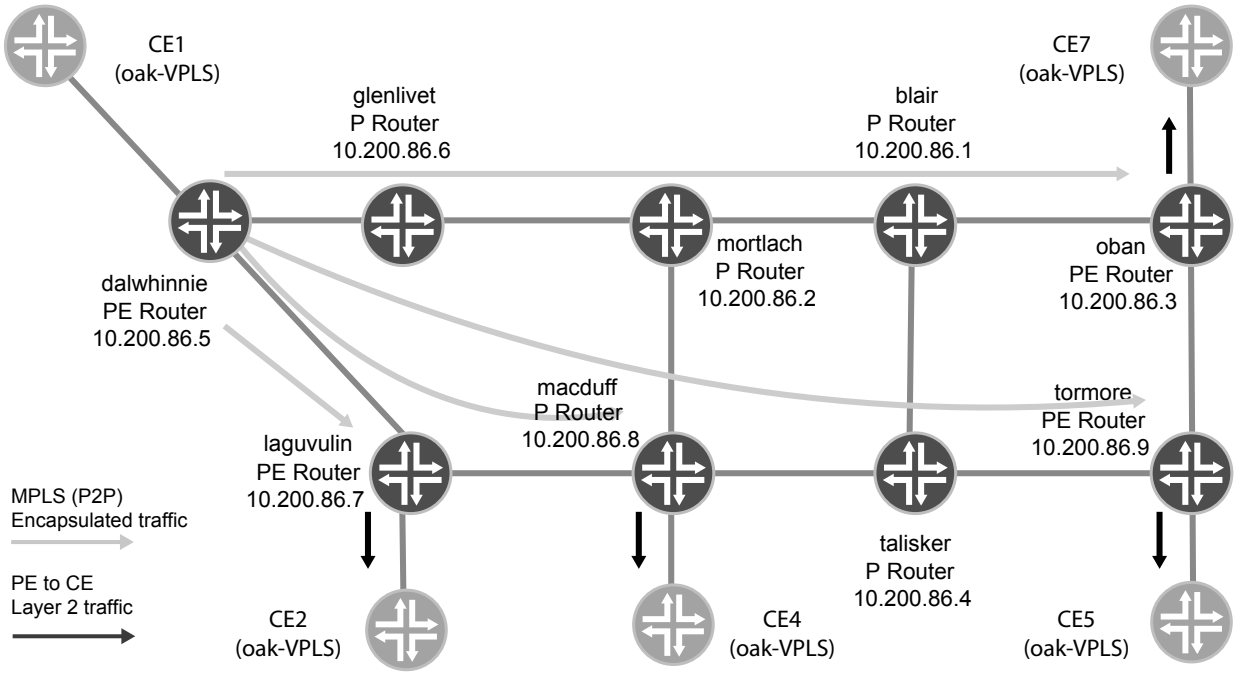



Figure 3.14 BUM Traffic Using P2P LSPs

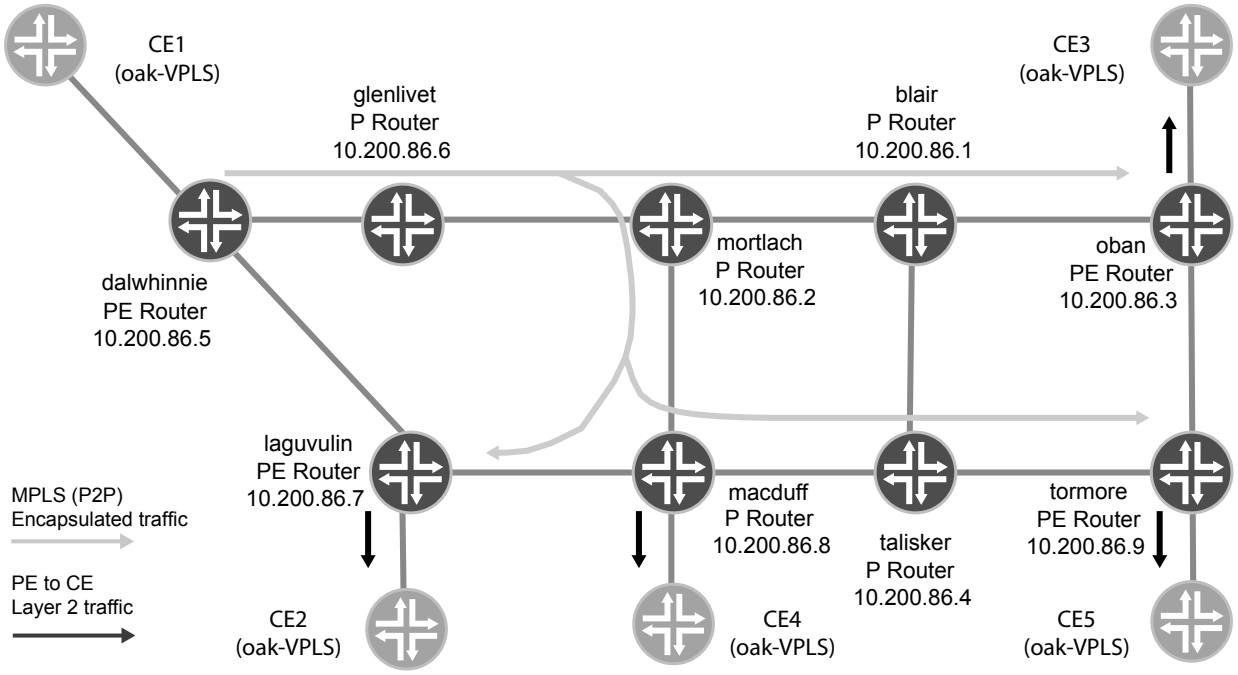



Figure 3.15 BUM Traffic Using P2MP LSPs

The two figures depict a five CE VPLS domain. CE1 is sourcing BUM traffic (an ARP in this case) that must be flooded to all other CEs in the VPLS domain. If this network were utilizing P2P LSPs, the source PE must replicate the packet four times, sending one copy over each P2P LSP. With P2MP LSPs, the source PE only sends a single copy of the packet into the P2MP LSP, and mortlach will replicate the packet once, sending it towards oban and macduff. Macduff will replicate the packet two times, sending one towards lagavulin, one towards tormore, and a final copy towards its local CE. P2MPs reduce the replication burden on the head-end PE and reduce the load on the network by removing unnecessary copies of the packet. Packets replicate only at branch points, which make this a far more efficient method of BUM traffic delivery.

For all that it accomplishes, the configuration of P2MP LSPs for VPLS is very simple. The configuration resides under the routing-instance hierarchy, such as the configuration here:

```
ps@dalwhinnie> show configuration routing-instances oak
instance-type vpls;
interface ge-1/0/0.0;
route-distinguisher 10.200.86.1:100;
provider-tunnel {
    rsvp-te {
        label-switched-path-template {
            default-template;
        }
    }
}
vrf-target target:300:200;
protocols {
    vpls {
        site-range 5;
        no-tunnel-services;
        site oak-ce1 {
            site-identifier 1;
            interface ge-1/0/0.0;
        }
    }
}
```

Note that it is not necessary to configure individual LSPs – the router will signal a P2MP LSP with branches to each PE participating in the VPLS instance. The name of the automatically signaled P2MP LSP trunk comes in the format of *route-distinguisher*:*instance name* and the branches have the loopback address of the destination router prepended to the trunk name. For example, the trunk P2MP LSP from dalwhinnie is named *10.200.86.1:100:vpls:oak* and the branches to macduff and lagavulin are 10.200.86.3:10.200.86.1:100:vpls:oak and 10.200.86.5:10.200.86.1:100:vpls:oak respectively.

The output of `show mpls lsp` and `show mpls lsp detail` shows these LSPs:

```
ps@dalwhinnie> show mpls lsp ingress
Ingress LSP: 3 sessions
To            From          State Rt P    ActivePath      LSPname
10.200.86.3    10.200.86.1    Up    0 *                   dalwhinnie-to-macduff
10.200.86.3    10.200.86.1    Up    0 *                   10.200.86.3:10.200.86.1:100:vpls:oak
10.200.86.5    10.200.86.1    Up    0 *                   dalwhinnie-to-lagavulin
10.200.86.5    10.200.86.1    Up    0 *                   10.200.86.5:10.200.86.1:100:vpls:oak
Total 4 displayed, Up 4, Down 0
```

The `detail` version of the `show mpls lsp` command provides additional information about the LSP, showing the values set by the default-template LSP. The bold portion of the output highlights the attributes set by the default-template:

```
ps@dalwhinnie> show mpls lsp detail name 10.200.86.3:10.200.86.1:100:vpls:o
ak
Ingress LSP: 3 sessions
10.200.86.3
From: 10.200.86.1, State: Up, ActiveRoute: 0, LSPname:
10.200.86.3:10.200.86.1:100:vpls:oak
ActivePath:  (primary)
P2MP name: 10.200.86.1:100:vpls:oak
PathDomain: Inter-domain
LSPtype: Dynamic Configured
LoadBalance: Random
Encoding type: Packet, Switching type: Packet, GPID: IPv4
*Primary                 State: Up
Priorities: 7 0
SmartOptimizeTimer: 180
```

If customized settings are required, a user-defined template can replace the default-template. The following template specifies a non-default optimize-timer:

```
ps@dalwhinnie> show configuration protocols mpls label-switched-path P2MP-template
template;
optimize-timer 35;
p2mp;
```

The configuration to apply this LSP to a VPLS routing-instance is very similar to the use of the default-template. Instead of specifying the default-template, the LSP name created is used:

```
ps@dalwhinnie> show configuration routing-instances oak
instance-type vpls;
interface ge-1/0/0.0;
route-distinguisher 10.200.86.1:100;
provider-tunnel {
    rsvp-te {
        label-switched-path-template {
            P2MP-template;
        }
    }
}
vrf-target target:300:200;
protocols {
    vpls {
        site-range 6;
        no-tunnel-services;
        site vpls1-ce1 {
            site-identifier 1;
            interface ge-1/0/0.0;
        }
    }
}
```

Following a commit, the LSP now shows the non-default optimize-timer configuration:

```
ps@dalwhinnie> show mpls lsp detail name 10.200.86.3:10.200.86.1:100:vpls:oak
Ingress LSP: 3 sessions
10.200.86.3
From: 10.200.86.1, State: Up, ActiveRoute: 0, LSPname: 10.200.86.3:10.200.86.1:100:vpls:oak
```

```
ActivePath:  (primary)
P2MP name: 10.200.86.1:100:vpls:oak
LSPtype: Dynamic Configured
LoadBalance: Random
Encoding type: Packet, Switching type: Packet, GPID: IPv4
*Primary                 State: Up
Priorities: 7 0
OptimizeTimer: 35
SmartOptimizeTimer: 180
```

Dalwhinnie now uses this (P2MP) LSP to deliver broadcast, unknown unicast, and multicast traffic. Note that unicast traffic uses a P2P LSP. A destination MAC is learned.

## Filtering BUM Traffic

A second method of scaling VPLS deployments is to filter the amount of Layer 2 broadcast, unknown unicast and multicast traffic. To accomplish this, the user configures a family VPLS firewall filter and applies this filter to the routing-instance at the Junos forwarding-options hierarchy.

A sample below shows a configuration utilizing a firewall filter within a routing-instance to protect the network against floods of BUM traffic (note that the policer values of 50 megabits need customization based on the needs of the VPLS instance, the CE to PE circuit, and the overall capacity of the network):

```
ps@dalwhinnie> show configuration firewall
family vpls {
    filter OakVplsBumFilter {
        term LimitBroadcast {
            from {
                traffic-type broadcast;
            }
            then {
                policer 50M-Policer;
                accept;
            }
        }
        term LimitMulticast {
            from {
                traffic-type multicast;
            }
            then {
                policer 50M-Policer;
                accept;
            }
        }
        term LimitUnknownUnicast {
            from {
                traffic-type unknown-unicast;
            }
            then {
                policer 50M-Policer;
                accept;
            }
        }
        term ExplicitPermit {
            then accept;
        }
    }
```

```
}
policer 50M-Policer {
    filter-specific;
    if-exceeding {
        bandwidth-limit 50m;
        burst-size-limit 10m;
    }
    then discard;
}
```

Finally, the routing-instance will include the firewall filter created above under the [forwarding-options] hierarchy, as shown here:

```
ps@dalwhinnie> show configuration routing-instances oak
instance-type vpls;
interface ge-1/0/0.0;
route-distinguisher 10.200.86.1:100;
provider-tunnel {
    rsvp-te {
        label-switched-path-template {
            default-template;
        }
    }
}
vrf-target target:300:200;
forwarding-options {
    family vpls {
        filter {
            input OakVplsBumFilter;
        }
    }
}
```

## Summary

MPLS is an exciting tool set that delivers features and functionality that either did not previously exist or had operational or scaling limitations. The seasoned engineer knows that nothing is free and with this functionality comes network overhead. It is critical to understand the scaling factors as well as the options for mitigating their impact when deploying and growing an MPLS network. The methods described in this section should give you the tools you need to plan your MPLS deployment and proactively manage its growth.

# Chapter 4

## MPLS Deployment Examples

Chapter 3 emphasized that the MPLS protocols and their implementation largely depend on the purpose and requirements of the network, so now let's examine several sample MPLS deployments, and focus on their network requirements as well as their resulting design and configurations. Four networks are presented. They range from a fairly simple LDP network to a complex, traffic-engineered, RSVP network supporting MPLS VPN services on the edge. Let's start with the basics and move our way to greater complexity (and enhanced productivity).

## A Basic Network

- IGP: OSPF
- IGP Scaling: None
- Label distribution protocol: LDP
- BGP: Full edge mesh, BGP-free core
- Growth expectation: Little to no growth
- Additional requirements: Support MPLS based feature growth

The requirement of the network in the basic design is simply to support additional feature growth in the future. One of these requirements is MPLS VPN services. Because this network is not expected to grow significantly, the network is deployed with a single OSPF area and LDP configured on all core and edge-facing interfaces. Its lack of traffic-engineering requirements, combined with the simplicity of LDP, makes the basic design the right label distribution protocol in this environment, as illustrated in Figure 4.1.

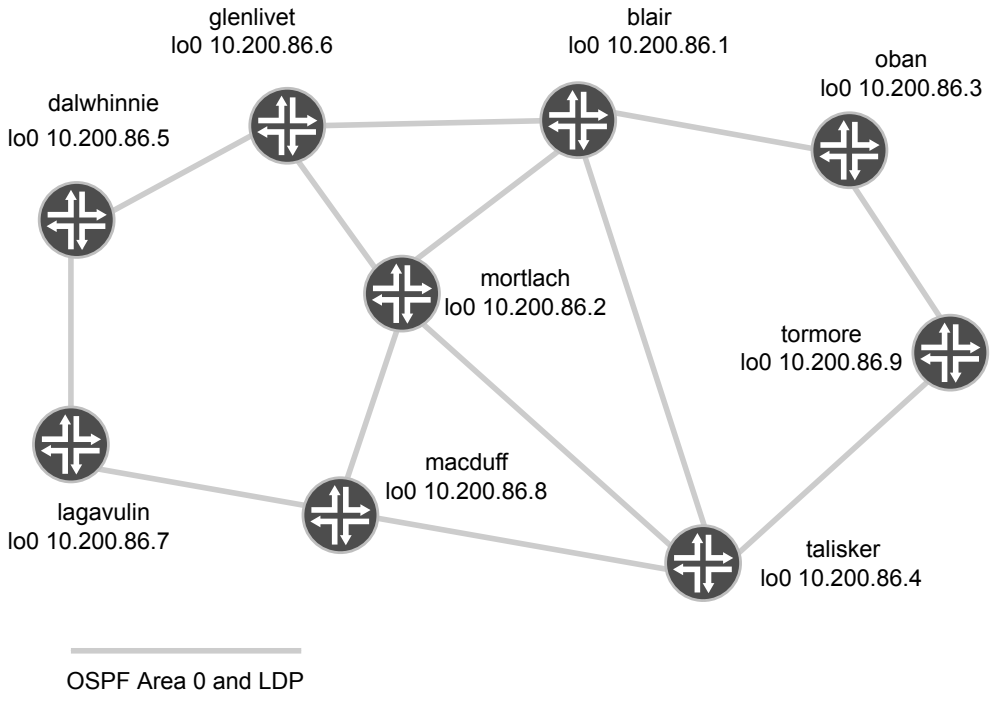



Figure 4.1 Basic Network Diagram

Figure 4.1 indeed shows a simple configuration. Each router has its loopback interfaces and core/edge facing interfaces configured for OSPF area 0. Because the physical interfaces are point-to-point links, they are configured with `interface-type p2p` to speed convergence. The PE routers oban, tormore, dalwinnie, and lagavulin are configured in a full BGP mesh. As this network is running LDP, there is no need for the core routers to be running BGP, which gives us the opportunity to implement a BGP-free core. The PE routers participate in BGP AS 1 and peer between loopback addresses. These routers exchange family inet, family inet-vpn and family l2vpn allowing for easy VPN implementation when that time comes.

The configurations to support this design fall into one of two categories, PE and P. Below, example PE and P configurations are shown in both Junos and Set Style.

## Dalwhinnie (PE)

Dalwhinie has its two interfaces configured for OSPF and LDP and is configured to peer with the other PE routers. These BGP sessions are configured with three different families to support the unicast inet prefixes exchanged today, as well as the inet-vpn unicast routes and l2vpn signaling (L2VPN and VPLS support) for future feature deployment.

### Junos Style

```
system {
    host-name dalwhinnie;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2003;
            class super-user;
            authentication {
                encrypted-password "$1$9iVZGMcI$Bz5/GWXsO32k1s2aOiEo70"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/2";
```

```
            family inet {
            address 192.168.86.6/30;
         }
         family mpls;
      }
   }
   ge-0/0/3 {
      unit 0 {
         description "Connection to lagavulin ge-0/0/3";
         family inet {
            address 192.168.86.30/30;
         }
         family mpls;
      }
   }
   lo0 {
      unit 0 {
         family inet {
            address 10.200.86.5/32;
         }
      }
   }
}
routing-options {
   router-id 10.200.86.5;
   autonomous-system 1;
}
protocols {
   bgp {
      group ibgp {
         type internal;
         local-address 10.200.86.5;
         family inet {
            unicast;
         }
         family inet-vpn {
            unicast;
         }
         family l2vpn {
            signaling;
         }
         neighbor 10.200.86.7;
         neighbor 10.200.86.9;
         neighbor 10.200.86.3;
      }
   }
   ospf {
      area 0.0.0.0 {
         interface lo0.0;
         interface ge-0/2/0.0;
         interface ge-0/3/0.0;
      }
   }
   ldp {
      interface ge-0/2/0.0;
      interface ge-0/3/0.0;
   }
}
```

### Set Style

```
set system host-name dalwhinnie
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."
set system login user ps uid 2003
set system login user ps class super-user
set system login user ps authentication encrypted-password
"$1$9iVZGMcI$Bz5/GWXsO32k1s2aOiEo70"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/2/0 unit 0 description "Connection to glenlivet ge-
0/2/0"
set interfaces ge-0/2/0 unit 0 family inet address 192.168.86.6/30
set interfaces ge-0/2/0 unit 0 family mpls
set interfaces ge-0/3/0 unit 0 description "Connection to lagavulin ge-
0/3/0"
set interfaces ge-0/3/0 unit 0 family inet address 192.168.86.30/30
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.5/32
set routing-options router-id 10.200.86.5
set routing-options autonomous-system 1
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.5
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.7
set protocols bgp group ibgp neighbor 10.200.86.9
set protocols bgp group ibgp neighbor 10.200.86.3
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/2/0.0
set protocols ospf area 0.0.0.0 interface ge-0/3/0.0
set protocols ldp interface ge-0/2/0.0
set protocols ldp interface ge-0/3/0.0
```

## Mortlach (P)

As a P router in this design, mortlach's configuration is quite simple: OSPF and LDP are configured on each of its interfaces. Because mortlach is a P router in a BGP-free core environment, it does not have any BGP sessions and switches packets based entirely on their MPLS labels.

### Junos Style

```
system {
   host-name mortlach;
   root-authentication {
      encrypted-password "$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2005;
         class super-user;
         authentication {
            encrypted-password "$1$2OY5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
         }
```

```
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        host 192.168.11.135 {
            any info;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
   fe-0/0/1 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/3";
            family inet {
                address 192.168.86.45/30;
            }
            family mpls;
        }
   }
   fe-2/0/0 {
        unit 0 {
            description "Connection to blair ge-0/0/2";
               family inet {
               address 192.168.86.49/30;
            }
            family mpls;
        }
    }
    e1-1/0/0 {
        unit 0 {
            description "Connection to talisker e1-1/0/0";
            family inet {
                address 192.168.86.22/30;
            }
            family mpls;
        }
    }
    fe-2/0/1 {
        unit 0 {
            description "Connection to macduff ge-0/0/3";
            family inet {
                address 192.168.86.41/30;
            }
            family mpls;
        }
   }
   lo0 {
        unit 0 {
            family inet {
                address 10.200.86.2/32 {
                    primary;
```

```
            }
         }
      }
   }
}
routing-options {
   router-id 10.200.86.2;
}
protocols {
   ospf {
      area 0.0.0.0 {
         interface lo0.0;
         interface fe-0/0/1.0;
         interface e1-1/0/0.0;
         interface fe-2/0/0.0;
         interface fe-2/0/1.0;
       }
   }
   ldp {
       interface fe-0/0/1.0;
       interface e1-1/0/0.0;
       interface fe-2/0/0.0;
       interface fe-2/0/1.0;
   }
}
```

### Set Style

```
set system host-name mortlach
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$2OY5E3.P$KCIi/
yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog host 192.168.11.135 any info
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces fe-0/0/1 unit 0 description "Connection to glenlivet ge-0/0/3"
set interfaces fe-0/0/1 unit 0 family inet address 192.168.86.45/30
set interfaces fe-0/0/1 unit 0 family mpls
set interfaces e1-1/0/0 unit 0 description "Connection to talisker e1-1/0/0"
set interfaces e1-1/0/0 unit 0 family inet address 192.168.86.22/30
set interfaces e1-1/0/0 unit 0 family mpls
set interfaces fe-2/0/0 unit 0 description "Connection to blair ge-0/0/2"
set interfaces fe-2/0/0 unit 0 family inet address 192.168.86.49/30
set interfaces fe-2/0/0 unit 0 family mpls
set interfaces fe-2/0/1 unit 0 description "Connection to macduff fe-2/0/1"
set interfaces fe-2/0/1 unit 0 family inet address 192.168.86.41/30
set interfaces fe-2/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.2/32 primary
set routing-options router-id 10.200.86.2
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface fe-0/0/1.0
set protocols ospf area 0.0.0.0 interface e1-1/0/0.0
set protocols ospf area 0.0.0.0 interface fe-2/0/0.0
set protocols ospf area 0.0.0.0 interface fe-2/0/1.0
set protocols ldp interface fe-0/0/1.0
set protocols ldp interface e1-1/0/0.0
set protocols ldp interface fe-2/0/0.0
set protocols ldp interface fe-2/0/1.0
```

## A Moderate Network

- IGP: IS-IS
- IGP Scaling: None
- Label distribution protocol: RSVP
- BGP: Full edge mesh, BGP-free core
- Additional requirements: Administrative group and static ERO based traffic engineering

In this slightly more complicated design, high bandwidth applications, such as periodic backup, require traffic engineering capabilities. Because this network is not expected to grow significantly, the network is deployed with a single IS-IS level. RSVP is configured on all core- and edge-facing interfaces and LSPs are configured between all PE routers to meet the traffic engineering requirements, as illustrated in Figure 4.2.

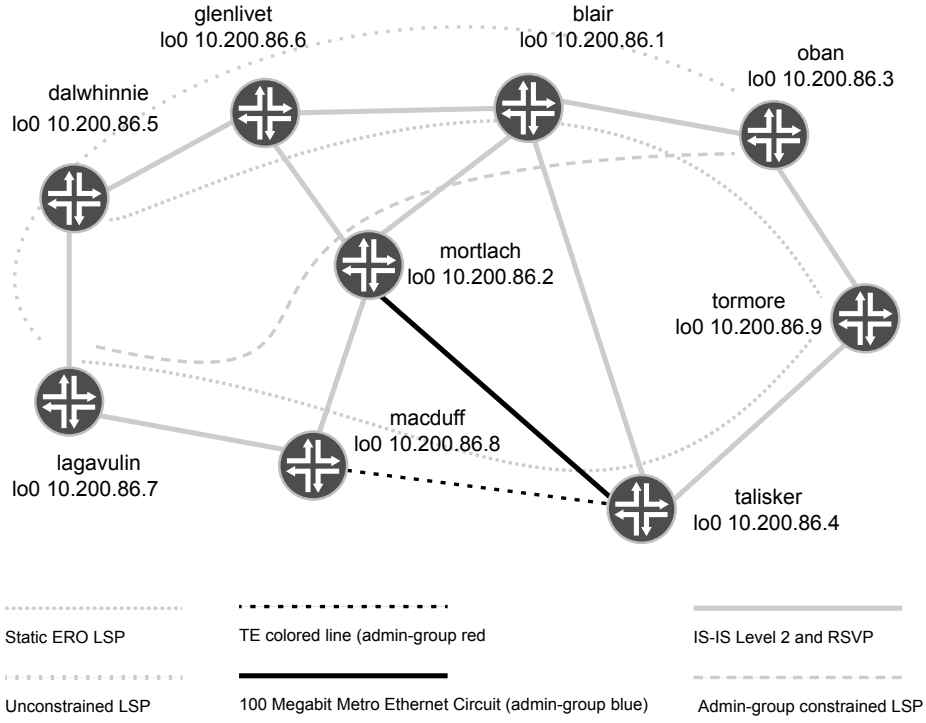



Figure 4.2 Moderate Network Design

The two dashed links in Figure 4.2 indicate a 100-megabit circuit (all of the other circuits are 1 gigabit). Tormore sinks a lot of traffic, spiking to over 500 megabits each from dalwhinnie and lagavulin, and as such, neither of these LSPs can traverse the 100-megabit links. Also note that they cannot be allowed to traverse the same link (except in failure mode). These LSPs will be configured with secondary LSPs to protect against failure of any of the links in the primary explicit path. While these secondary LSPs may cause an overutilization, the network is not rich enough to provide a fully-redundant alternate path. The lagavulin to oban LSP frequently transmits at up to 150 megabits and as a result, this LSP cannot traverse the 100-megabit links. Because this LSP, plus either of the static ERO LSPs (totaling 650

megabits), can share any of the 1-gigabit links, all you need to guard against is that this LSP does not traverse the 100-megabit links – a good use of administrative groups.

To ensure the desired traffic-engineering, *color* the 100-megabit links blue and instruct the lagavulin to oban LSP to avoid blue links. To prevent the dalwhinnie to tormore LSP from traversing the same links, the macduff to talisker and talisker to tormore links are colored *red* and the dalwhinnie to tormore LSP has been configured to avoid red links. This should cause this LSP to take the path shown in the diagram (note that in some editions of this book color is shown as solid, dotted, or gray lines). The remaining LSPs do not carry significant traffic and are therefore configured without any constraints. Because of the complexity, the configurations for dalwhinnie, lagavulin, glenlivet, and macduff are detailed here. Please note that traffic-engineering is not a required statement for RSVP signaled LSPs when using IS-IS, because IS-IS carries the traffic-engineering TLVs.

## Dalwhinnie (PE)

Most of the interesting configuration pieces on dalwhinnie are within the [protocols mpls] hierarchy. Note that there are two LSPs from dalwhinnie to tormore, the first (-primary) is configured with the admin-group constraints forcing the LSP onto the path identified and the second (-secondary) is configured without any constraints, allowing it to take the shortest available path. The `metric` keyword is used to prefer the primary LSP over the secondary.

### Junos Style

```
system {
   host-name dalwhinnie;
   root-authentication {
      encrypted-password "$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2003;
         class super-user;
         authentication {
            encrypted-password "$1$9iVZGMcI$Bz5/GWXsO32k1s2a0iEo70"; ##
SECRET-DATA
         }
      }
   }
   services {
      ssh;
   }
   syslog {
      user * {
         any emergency;
      }
      file messages {
         any notice;
         authorization info;
      }
      file interactive-commands {
         interactive-commands any;
      }
   }
```

```
}
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/2";
                family inet {
                address 192.168.86.6/30;
            }
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to lagavulin ge-0/0/3";
            family inet {
                address 192.168.86.30/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.5/32;
            }
            family iso {
                address 49.0001.1000.0000.0001.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.5;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface ge-0/0/2.0;
        interface ge-0/0/2.0;
    }
    mpls {
        admin-groups {
            blue 0;
            red 1;
        }
        label-switched-path dalwhinnie-to-oban {
            to 10.200.86.3;
        }
        label-switched-path dalwhinnie-to-lagavulin {
            to 10.200.86.7;
        }
        label-switched-path dalwhinnie-to-tormore-primary {
            to 10.200.86.9;
            metric 100;
            admin-group exclude [ blue red ];
        }
        label-switched-path dalwhinnie-to-tormore-secondary {
            to 10.200.86.9;
            metric 200;
        }
        interface ge-0/0/2.0;
```

```
        interface ge-0/0/3.0;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.5;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 10.200.86.7;
            neighbor 10.200.86.9;
            neighbor 10.200.86.3;
        }
    }
    isis {
        level 1 disable;
        level 2 wide-metrics-only;
        interface ge-0/0/2.0 {
            level 1 disable;
        }
        interface ge-0/0/3.0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
}
```

### Set Style

```
set system host-name dalwhinnie
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2003
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$9iVZGMcI$Bz5/
GWXsO32k1s2aOiEo70"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/2/0 unit 0 description "Connection to glenlivet ge-0/2/0"
set interfaces ge-0/2/0 unit 0 family inet address 192.168.86.6/30
set interfaces ge-0/2/0 unit 0 family mpls
set interfaces ge-0/3/0 unit 0 description "Connection to lagavulin ge-0/3/0"
set interfaces ge-0/3/0 unit 0 family inet address 192.168.86.30/30
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.5/32
set interfaces lo0 unit 0 family iso address 49.0001.1000.0000.0001.00
set routing-options router-id 10.200.86.5
set routing-options autonomous-system 1
set protocols rsvp interface ge-0/0/2.0
set protocols rsvp interface ge-0/0/3.0
```

```
set protocols mpls admin-groups blue 0
set protocols mpls admin-groups red 1
set protocols mpls label-switched-path dalwhinnie-to-oban to 10.200.86.3
set protocols mpls label-switched-path dalwhinnie-to-lagavulin to
10.200.86.7
set protocols mpls label-switched-path dalwhinnie-to-tormore-primary to
10.200.86.9
set protocols mpls label-switched-path dalwhinnie-to-tormore-primary metric
100
set protocols mpls label-switched-path dalwhinnie-to-tormore-primary
admin-group exclude blue
set protocols mpls label-switched-path dalwhinnie-to-tormore-primary
admin-group exclude red
set protocols mpls label-switched-path dalwhinnie-to-tormore-secondary to
10.200.86.9
set protocols mpls label-switched-path dalwhinnie-to-tormore-secondary
metric 200
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.5
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.7
set protocols bgp group ibgp neighbor 10.200.86.9
set protocols bgp group ibgp neighbor 10.200.86.3
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface ge-0/0/2.0 level 1 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
```

## Lagavulin (PE)

Similar to dalwhinnie, the interesting portions of the lagavulin configuration are found in the [protocols mpls] section. Lagavulin has a similar LSP configuration for the tormore LSP. Additionally, the lagavulin-to-oban LSP has two paths, a primary which specifies mortlach as a loose hop and a secondary with no path constraints. This secondary is configured with the `standby` knob, which allows the LSP to be pre-signaled and expedites failover.

### Junos Style

```
system {
   host-name lagavulin;
   root-authentication {
      encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86YhU7CpvPe0"; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2005;
         class super-user;
         authentication {
            encrypted-password "$1$2OY5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
         }
      }
   }
```

```
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to macduff ge-0/0/2";
            family inet {
                address 192.168.86.2/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to dalwhinnie ge-0/0/3";
            family inet {
                address 192.168.86.29/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.7/32 {
                    primary;
                }
            }
            family iso {
                address 49.0001.1001.0000.0001.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.7;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface all;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    mpls {
```

```
        admin-groups {
            blue 0;
            red 1;
        }
        label-switched-path lagavulin-to-dalwhinnie {
            to 10.200.86.5;
        }
        label-switched-path lagavulin-to-tormore-primary {
            to 10.200.86.9;
            admin-group exclude blue;
        }
        label-switched-path lagavulin-to-tormore-secondary {
            to 10.200.86.9;
            metric 200;
        }
        label-switched-path lagavulin-to-oban {
            to 10.200.86.3;
            primary to-oban-primary;
            secondary to-oban-secondary {
                standby;
            }
        }
        path to-oban-primary {
            10.100.7.21 loose;
        }
        path to-oban-secondary;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.7;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 10.200.86.5;
            neighbor 10.200.86.3;
            neighbor 10.200.86.9;
        }
    }
    isis {
        level 1 disable;
        level 2 wide-metrics-only;
        interface ge-0/0/2.0 {
            level 1 disable;
        }
        interface ge-0/0/3.0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
}
```

### Set Style

```
set system host-name lagavulin
set system root-authentication encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86Yh
U7CpvPe0"
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$20Y5E3.P$KCIi/
yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/2 unit 0 description "Connection to macduff ge-0/0/2"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.2/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to dalwhinnie ge-0/0/3"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.29/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.7/32 primary
set interfaces lo0 unit 0 family iso address 49.0001.1000.0000.0001.00
set routing-options router-id 10.200.86.7
set routing-options autonomous-system 1
set protocols rsvp interface all
set protocols rsvp interface ge-0/0/2.0
set protocols rsvp interface ge-0/0/3.0
set protocols mpls admin-groups blue 0
set protocols mpls admin-groups red 1
set protocols mpls label-switched-path lagavulin-to-dalwhinnie to 10.200.86.5
set protocols mpls label-switched-path lagavulin-to-tormore-primary to
10.200.86.9
set protocols mpls label-switched-path lagavulin-to-tormore-primary admin-
group exclude blue
set protocols mpls label-switched-path lagavulin-to-oban to 10.200.86.3
set protocols mpls label-switched-path lagavulin-to-oban primary to-oban-
primary
set protocols mpls label-switched-path lagavulin-to-oban secondary to-oban-
secondary standby
set protocols mpls label-switched-path lagavulin-to-tormore-secondary to
10.200.86.9
set protocols mpls label-switched-path lagavulin-to-tormore-secondary metric
200
set protocols mpls path to-oban-primary 10.100.7.21 loose
set protocols mpls path to-oban-secondary
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.7
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.5
set protocols bgp group ibgp neighbor 10.200.86.3
set protocols bgp group ibgp neighbor 10.200.86.9
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface ge-0/0/2.0 level 1 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
```

## Glenlivet (P)

The P routers provide an example of configuring a particular link with an admin-group. In the [protocols mpls] hierarchy, ge-0/0/1 is *colored blue*, which the PEs are configured to avoid on the tormore LSPs.

### Junos Style

```
system {
    host-name glenlivet;
    root-authentication {
        encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86YhU7CpvPe0"; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2005;
            class super-user;
            authentication {
                encrypted-password "$1$20Y5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/1 {
        unit 0 {
            description "Connection to blair ge-0/0/1";
            family inet {
                address 192.168.86.10/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/2 {
        unit 0 {
            description "Connection to dalwhinnie ge-0/0/2";
            family inet {
                address 192.168.86.5/30;
            }
            family iso;
            family mpls;
        }
```

```
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to mortlach fe-0/0/1";
            family inet {
                address 192.168.86.46/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.6/32;
            }
            family iso {
                address 49.0001.1000.0000.0002.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.6;
}
protocols {
    rsvp {
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    mpls {
        admin-groups {
            blue 0;
            red 1;
        }
        interface ge-0/0/1.0 {
            admin-group blue;
        }
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    isis {
        level 1 disable;
        level 2 wide-metrics-only;
        interface ge-0/0/1.0 {
            level 1 disable;
        }
        interface ge-0/0/2.0 {
            level 1 disable;
        }
        interface ge-0/0/3.0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
}
```

### Set Style

```
set system host-name glenlivet
set system root-authentication encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86
YhU7CpvPe0"
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$20Y5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/1 unit 0 description "Connection to blair ge-0/0/1"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.86.10/30
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 description "Connection to dalwhinnie ge-
0/0/2"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.5/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to mortlach fe-0/0/1"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.46/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.6/32
set interfaces lo0 unit 0 family iso address 49.0001.1002.0000.0001.00
set routing-options router-id 10.200.86.6
set protocols rsvp interface ge-0/0/1.0
set protocols rsvp interface ge-0/0/2.0
set protocols rsvp interface ge-0/0/3.0
set protocols mpls admin-groups blue 0
set protocols mpls admin-groups red 1
set protocols mpls interface ge-0/0/1.0 admin-group blue
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface ge-0/0/1.0 level 1 disable
set protocols isis interface ge-0/0/2.0 level 1 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
```

## Macduff (P)

Again, the P routers show use of link-coloring. In the [protocols mpls] hierarchy, ge-0/0/1.0 is *colored red*, which dalwhinnie is configured to avoid on its tormore LSP (in addition to *blue links*).

### Junos Style

```
system {
   host-name macduff;
   root-authentication {
      encrypted-password "$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2005;
```

```
            class super-user;
            authentication {
               encrypted-password "$1$20Y5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
            }
         }
      }
      services {
         ssh;
      }
      syslog {
         user * {
            any emergency;
         }
         file messages {
            any notice;
            authorization info;
         }
         file interactive-commands {
            interactive-commands any;
         }
      }
}
interfaces {
   ge-0/0/1 {
      unit 0 {
         description "Connection to talisker fe-0/0/1";
         family inet {
            address 192.168.86.14/30;
         }
         family iso;
         family mpls;
      }
   }
   ge-0/0/2 {
      unit 0 {
         description "Connection to lagavulin ge-0/0/2";
         family inet {
            address 192.168.86.1/30;
         }
         family iso;
         family mpls;
      }
   }
   ge-0/0/3 {
      unit 0 {
         description "Connection to mortlach fe-2/0/1";
         family inet {
            address 192.168.86.42/30;
         }
         family iso;
         family mpls;
      }
   }
   lo0 {
      unit 0 {
         family inet {
            address 10.200.86.8/32 {
               primary;
            }
         }
         family iso {
            address 49.0001.1003.0000.0001.00;
```

```
            }
        }
    }
}
routing-options {
    router-id 10.200.86.8;
}
protocols {
    rsvp {
        interface all;
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    mpls {
        admin-groups {
            blue 0;
            red 1;
        }
        interface ge-0/0/1.0 {
            admin-group red;
        }
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    isis {
        level 1 disable;
        level 2 wide-metrics-only;
        interface ge-0/0/1.0 {
            level 1 disable;
        }
        interface ge-0/0/2.0 {
            level 1 disable;
        }
        interface ge-0/0/3.0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
}
```

### Set Style

```
set system host-name macduff
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$2OY5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/1 unit 0 description "Connection to talisker fe-0/0/1"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.86.14/30
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
```

```
set interfaces ge-0/0/2 unit 0 description "Connection to lagavulin ge-0/0/2"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.1/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to mortlach fe-2/0/1"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.42/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.8/32 primary
set interfaces lo0 unit 0 family iso address 49.0001.1003.0000.0001.00
set routing-options router-id 10.200.86.8
set protocols rsvp interface all
set protocols rsvp interface ge-0/0/1.0
set protocols rsvp interface ge-0/0/2.0
set protocols rsvp interface ge-0/0/3.0
set protocols mpls admin-groups blue 0
set protocols mpls admin-groups red 1
set protocols mpls interface ge-0/0/1.0 admin-group red
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface ge-0/0/1.0 level 1 disable
set protocols isis interface ge-0/0/2.0 level 1 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
```

## A Moderate Network Evolved

- IGP: IS-IS
- IGP Scaling: Multi-area/level
- Label distribution protocol: LDP and RSVP
- BGP: Route-reflection
- Growth expectation: Moderate growth
- Additional requirements: Auto-bandwidth on RSVP signaled LSPs

The moderate network deployed in the previous example is experiencing unexpected growth and the network needs to scale to meet this demand. The growth includes additional PE devices, and as a result, more devices participating in level 2 IS-IS, more BGP peers, and more LSPs. To support the additional PEs, the network will be migrated to a multi-level design in which the PE routers reside in level 1 and the full edge BGP mesh are transitioned to a route-reflection model with dedicated route-reflectors that are not be in the forwarding plane. Finally, to allow for MPLS scaling, the edge routers are migrated to LDP with the core routers running a full LSP mesh. LDP is tunneled within these RSVP signaled LSPs, providing edge-to-edge MPLS connectivity. Figure 4.3 illustrates the design.

Because the operational aspect of manually managing LSPs becomes nearly impossible with so many LSPs and constraints, this network utilizes the Junos auto-bandwidth feature to reserve the appropriate bandwidth along paths which can provide the requested throughput. Also, note that the 100-megabit links have been upgraded to full 1-gigabit links to support the network growth. Examples of the configurations of the PE and P routers, as well as the route-reflectors, are provided here by way of dalwhinnie, oban, blair, glenlivet, and aberlour.

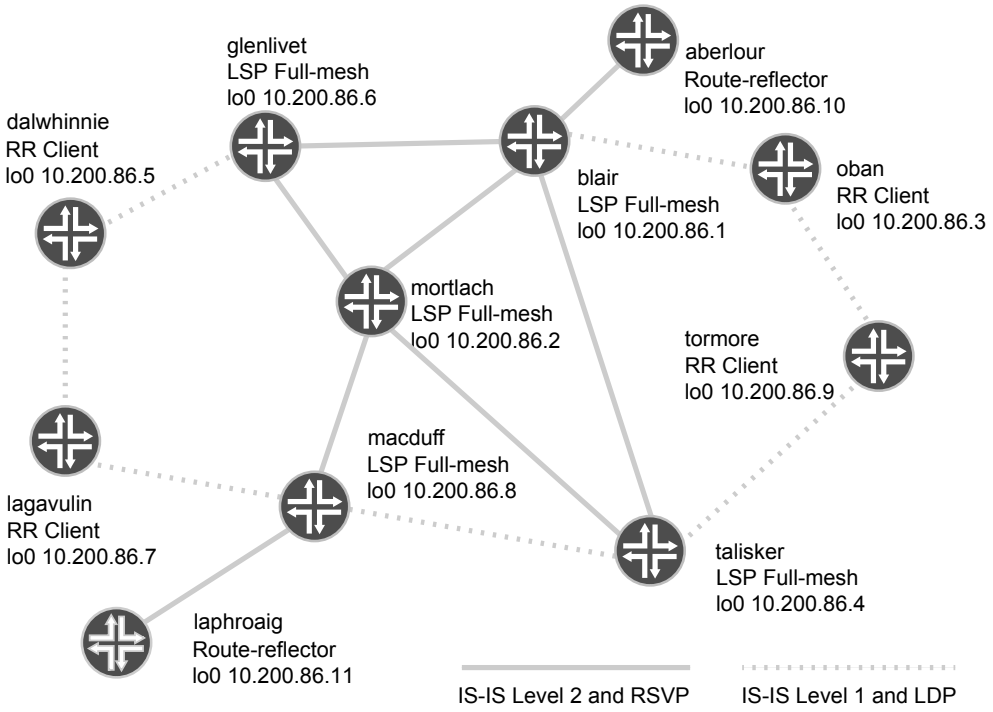



Figure 4.3 Moderate Network Evolved

## Dalwhinnie (PE)

Dalwhinnie is configured with LDP only on the uplink to glenlivet and the cross-connect to lagavulin. These links are also running in IS-IS level 1, area 49.0001. It is also a route-reflector client of the two route-reflectors. All four PE routers follow this design.

### Junos Style

```
system {
    host-name dalwhinnie;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2003;
            class super-user;
            authentication {
                encrypted-password "$1$9iVZGMcI$Bz5/GWXsO32k1s2a0iEo70"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
```

```
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/2";
                family inet {
                address 192.168.86.6/30;
            }
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to lagavulin ge-0/0/3";
            family inet {
                address 192.168.86.30/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.5/32;
            }
            family iso {
                address 49.0001.0102.0008.6005.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.5;
    autonomous-system 1;
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.5;
            neighbor 10.200.86.10;
            neighbor 10.200.86.11;
        }
    }
    isis {
        level 1 wide-metrics-only;
        level 2 disable;
        interface ge-0/.500 {
            level 2 disable;
        }
        interface ge-0/0/2.0 {
            level 2 disable;
        }
```

```
        interface ge-0/0/3.0 {
            level 2 disable;
        }
    }
    ldp {
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
}
```

Set Style

```
set system host-name dalwhinnie
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."
set system login user ps uid 2003
set system login user ps class super-user
set system login user ps authentication encrypted-password
"$1$9iVZGMcI$Bz5/GWXsO32k1s2a0iEo70"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/2/0 unit 0 description "Connection to glenlivet ge-
0/2/0"
set interfaces ge-0/2/0 unit 0 family inet address 192.168.86.6/30
set interfaces ge-0/2/0 unit 0 family mpls
set interfaces ge-0/3/0 unit 0 description "Connection to lagavulin ge-
0/3/0"
set interfaces ge-0/3/0 unit 0 family inet address 192.168.86.30/30
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.5/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.0008.6005.00
set routing-options router-id 10.200.86.5
set routing-options autonomous-system 1
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.5
set protocols bgp group ibgp neighbor 10.200.86.10
set protocols bgp group ibgp neighbor 10.200.86.11
set protocols isis level 1 wide-metrics-only
set protocols isis level 2 disable
set protocols isis interface ge-0/0/2.0 level 2 disable
set protocols isis interface ge-0/0/3.0 level 2 disable
set protocols isis interface lo0.0 level 2 disable
set protocols ldp interface ge-0/0/3.0
set protocols ldp interface ge-0/0/3.0
```

## Glenlivet (P)

The P routers are where things start getting more interesting. The P router IS-IS configuration allows for Level 2 routes to be leaked into Level 1. This is necessary for the PEs to install the IS-IS routes and associated LDP labels, otherwise, they would only have the default route installed based on the attached bit from their L1-L2 routers (their upstream P routers). Additionally, under the Junos [protocols mpls] stanza, a statistics file has been configured and auto-bandwidth data collection has been enabled. These statistics are gathered every 300 seconds by default, but other values can be configured. Finally, each LSP has been configured with ldp-tunneling as well as auto-bandwidth with an adjustment-interval of 3600

seconds (one hour). These statements instruct the router to exchange LDP labels with the router on the remote side of the LSP (ldp-tunneling), and also to reserve bandwidth based on the average bandwidth utilization of the LSP over the last 3600 seconds. When that timer expires, the LSP is re-signaled with an updated bandwidth reservation.

### Junos Style

```
system {
   host-name glenlivet;
   root-authentication {
      encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86YhU7CpvPe0"; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2005;
         class super-user;
         authentication {
            encrypted-password "$1$20Y5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
         }
      }
   }
   services {
      ssh;
   }
   syslog {
      user * {
         any emergency;
      }
      file messages {
         any notice;
         authorization info;
      }
      file interactive-commands {
         interactive-commands any;
      }
   }
}
interfaces {
  ge-0/0/1 {
      unit 0 {
         description "Connection to blair ge-0/0/1";
         family inet {
            address 192.168.86.10/30;
         }
         family iso;
         family mpls;
      }
   }
   ge-0/0/2 {
      unit 0 {
         description "Connection to dalwhinnie ge-0/0/2";
         family inet {
            address 192.168.86.5/30;
         }
         family iso;
         family mpls;
      }
   }
```

```
    ge-0/0/3 {
        unit 0 {
            description "Connection to mortlach fe-0/0/1";
            family inet {
                address 192.168.86.46/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.6/32;
            }
            family iso {
                address 49.0001.0102.0008.6006.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.6;
}
protocols {
    rsvp {
        interface ge-0/0/1.0;
        interface ge-0/0/3.0;
    }
    mpls {
        statistics {
            file mpls-stats;
            auto-bandwidth;
        }
        label-switched-path glenlivet-to-mortlach {
            to 10.200.86.2;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path glenlivet-to-talisker {
            to 10.200.86.4;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path glenlivet-to-macduff {
            to 10.200.86.8;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path glenlivet-to-blair {
            to 10.200.86.1;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        interface ge-0/0/1.0;
```

```
        interface ge-0/0/3.0;
    }
    isis {
        export PE-Loopbacks-to-Level-1;
        level 2 wide-metrics-only;
        level 1 wide-metrics-only;
        interface ge-0/0/1.0 {
            level 1 disable;
        }
        interface ge-0/0/2.0 {
            level 2 disable;
        }
        interface ge-0/0/3.0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
    ldp {
        interface ge-0/0/2.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement PE-Loopbacks-to-Level-1 {
        term Accept-Loopbacks {
            from {
                protocol isis;
                route-filter 10.200.86.0/24 prefix-length-range /32-/32;
            }
            then accept;
        }
        term Accept-Direct {
            from protocol direct;
            then accept;
        }
        term Reject-All-Else {
            then reject;
        }
    }
}
```

### Set Style

```
set system host-name glenlivet
set system root-authentication encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86Y
hU7CpvPe0"
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$20Y5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/1 unit 0 description "Connection to blair ge-0/0/1"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.86.10/30
```

```
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 description "Connection to dalwhinnie ge-
0/0/2"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.5/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to mortlach fe-0/0/1"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.46/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.6/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.0008.6006.00
set routing-options router-id 10.200.86.6
set protocols rsvp interface ge-0/0/1.0
set protocols rsvp interface ge-0/0/3.0
set protocols mpls statistics file mpls-stats
set protocols mpls statistics auto-bandwidth
set protocols mpls label-switched-path glenlivet-to-mortlach to 10.200.86.2
set protocols mpls label-switched-path glenlivet-to-mortlach ldp-tunneling
set protocols mpls label-switched-path glenlivet-to-mortlach auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path glenlivet-to-talisker to 10.200.86.4
set protocols mpls label-switched-path glenlivet-to-talisker ldp-tunneling
set protocols mpls label-switched-path glenlivet-to-talisker auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path glenlivet-to-macduff to 10.200.86.8
set protocols mpls label-switched-path glenlivet-to-macduff ldp-tunneling
set protocols mpls label-switched-path glenlivet-to-macduff auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path glenlivet-to-blair to 10.200.86.1
set protocols mpls label-switched-path glenlivet-to-blair ldp-tunneling
set protocols mpls label-switched-path glenlivet-to-blair auto-bandwidth
adjust-interval 3600
set protocols mpls interface ge-0/0/1.0
set protocols mpls interface ge-0/0/3.0
set protocols isis export PE-Loopbacks-to-Level-1
set protocols isis level 2 wide-metrics-only
set protocols isis level 1 wide-metrics-only
set protocols isis interface ge-0/0/1.0 level 1 disable
set protocols isis interface ge-0/0/2.0 level 2 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
set protocols ldp interface ge-0/0/2.0
set protocols ldp interface lo0.0
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks from protocol isis
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks from route-filter 10.200.86.0/24 prefix-length-range /32-/32
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks then accept
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Direct from protocol direct
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Direct then accept
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Reject-
All-Else then reject
```

## Oban (PE)

Oban's configuration is provided for completeness, but looks very similar to that of dalwhinnie.

### Junos Style

```
system {
    host-name oban;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2005;
            class super-user;
            authentication {
                encrypted-password "$1$20Y5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to blair ge-6/0/0";
            family inet {
                address 192.168.86.25/30;
            }
            family iso;
            family mpls;
        }
    ge-0/0/3 {
        unit 0 {
            description "Connection to tormore ge-0/0/3";
            family inet {
                address 192.168.86.38/30;
            }
            family iso;
            family mpls;
        }
    }
```

```
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.3/32;
            }
            family iso {
                address 49.0002.0102.0008.6003.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.3;
    autonomous-system 1;
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.3;
            neighbor 10.200.86.10;
            neighbor 10.200.86.11;
        }
    }
    isis {
        level 2 disable;
        level 1 wide-metrics-only;
        interface ge-0/0/2.0 {
            level 2 disable;
        }
        interface ge-0/0/3.0 {
            level 2 disable;
        }
        interface lo0.0 {
            level 2 disable;
        }
    }
    ldp {
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
}
```

### Set Style

```
set system host-name oban
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$2OY5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/2 unit 0 description "Connection to blair ge-6/0/0"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.25/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
```

```
set interfaces ge-0/0/3 unit 0 description "Connection to tormore ge-0/0/3"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.38/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.3/32
set interfaces lo0 unit 0 family iso address 49.0002.0102.0008.6003.00
set routing-options router-id 10.200.86.3
set routing-options autonomous-system 1
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.3
set protocols bgp group ibgp neighbor 10.200.86.10
set protocols bgp group ibgp neighbor 10.200.86.11
set protocols isis level 2 disable
set protocols isis level 1 wide-metrics-only
set protocols isis interface ge-0/0/2.0 level 2 disable
set protocols isis interface ge-0/0/3.0 level 2 disable
set protocols isis interface lo0.0 level 2 disable
set protocols ldp interface ge-0/0/2.0
set protocols ldp interface ge-0/0/3.0
```

## Blair (P)

Blair's configuration looks very similar to glenlivet's. The major difference is that blair is in IS-IS area 49.0002, which puts it in a different area then glenlivet (49.0001).

### Junos Style

```
system {
    host-name blair;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2005;
            class super-user;
            authentication {
                encrypted-password "$1$2OY5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
```

```
interfaces {
    ge-0/0/0 {
        unit 0 {
            description "Connection to aberlour ge-0/0/0";
            family inet {
                address 192.168.86.53/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/1 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/1";
            family inet {
                address 192.168.86.9/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/2 {
        unit 0 {
            description "Connection to mortlach fe-2/0/0";
            family inet {
                address 192.168.86.50/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to talisker fe-2/0/0";
            family inet {
                address 192.168.86.18/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-6/0/0 {
        unit 0 {
            description "Connection to oban ge-0/0/2";
            family inet {
                address 192.168.86.26/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.1/32 {
                    primary;
                }
            }
            family iso {
                address 49.0002.0102.0008.6001.00;
            }
        }
    }
```

```
}
routing-options {
    router-id 10.200.86.1;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    mpls {
        statistics {
            file mpls-stats;
            auto-bandwidth;
        }
        label-switched-path blair-to-glenlivet {
            to 10.200.86.6;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path blair-to-mortlach {
            to 10.200.86.2;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path blair-to-talisker {
            to 10.200.86.4;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        label-switched-path blair-to-macduff {
            to 10.200.86.8;
            ldp-tunneling;
            auto-bandwidth {
                adjust-interval 3600;
            }
        }
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    isis {
        export PE-Loopbacks-to-Level-1;
        level 2 wide-metrics-only;
        level 1 wide-metrics-only;
        interface ge-0/0/0.0 {
            level 2 disable;
        }
        interface ge-0/0/1.0 {
            level 1 disable;
        }
        interface ge-0/0/2.0 {
            level 1 disable;
        }
        interface ge-0/0/3.0 {
```

```
            level 1 disable;
        }
        interface ge-6/0/0.0 {
            level 2 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
    ldp {
        interface ge-6/0/0.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement PE-Loopbacks-to-Level-1 {
        term Accept-Loopbacks {
            from {
                protocol isis;
                route-filter 10.200.86.0/24 prefix-length-range /32-/32;
            }
            then accept;
        }
        term Accept-Direct {
            from protocol direct;
            then accept;
        }
        term Reject-All-Else {
            then reject;
        }
    }
}
```

Set Style

```
set system host-name blair
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$2OY5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/0 unit 0 description "Connection to aberlour ge-0/0/0"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.86.53/30
set interfaces ge-0/0/0 unit 0 family iso
set interfaces ge-0/0/1 unit 0 description "Connection to glenlivet ge-
0/0/1"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.86.9/30
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 description "Connection to mortlach fe-2/0/0"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.5O/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to talisker fe-2/0/0"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.18/30
set interfaces ge-0/0/3 unit 0 family iso
```

```
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces ge-6/0/0 unit 0 description "Connection to oban ge-0/0/2"
set interfaces ge-6/0/0 unit 0 family inet address 192.168.86.26/30
set interfaces ge-6/0/0 unit 0 family iso
set interfaces ge-6/0/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.1/32 primary
set interfaces lo0 unit 0 family iso address 49.0002.0102.0008.6001.00
set routing-options router-id 10.200.86.1
set routing-options autonomous-system 1
set protocols rsvp interface ge-0/0/1.0
set protocols rsvp interface ge-0/0/2.0
set protocols rsvp interface ge-0/0/3.0
set protocols mpls statistics file mpls-stats
set protocols mpls statistics auto-bandwidth
set protocols mpls label-switched-path blair-to-glenlivet to 10.200.86.6
set protocols mpls label-switched-path blair-to-glenlivet ldp-tunneling
set protocols mpls label-switched-path blair-to-glenlivet auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path blair-to-mortlach to 10.200.86.2
set protocols mpls label-switched-path blair-to-mortlach ldp-tunneling
set protocols mpls label-switched-path blair-to-mortlach auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path blair-to-talisker to 10.200.86.4
set protocols mpls label-switched-path blair-to-talisker ldp-tunneling
set protocols mpls label-switched-path blair-to-talisker auto-bandwidth
adjust-interval 3600
set protocols mpls label-switched-path blair-to-macduff to 10.200.86.8
set protocols mpls label-switched-path blair-to-macduff ldp-tunneling
set protocols mpls label-switched-path blair-to-macduff auto-bandwidth
adjust-interval 3600
set protocols mpls interface ge-0/0/1.0
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols isis export PE-Loopbacks-to-Level-1
set protocols isis level 2 wide-metrics-only
set protocols isis level 1 wide-metrics-only
set protocols isis interface ge-0/0/0.0 level 1 disable
set protocols isis interface ge-0/0/1.0 level 1 disable
set protocols isis interface ge-0/0/2.0 level 1 disable
set protocols isis interface ge-0/0/3.0 level 1 disable
set protocols isis interface ge-6/0/0.0 level 2 disable
set protocols isis interface lo0.0 level 1 disable
set protocols ldp interface ge-6/0/0.0
set protocols ldp interface lo0.0
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks from protocol isis
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks from route-filter 10.200.86.0/24 prefix-length-range /32-/32
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Loopbacks then accept
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Direct from protocol direct
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Accept-
Direct then accept
set policy-options policy-statement PE-Loopbacks-to-Level-1 term Reject-All-
Else then reject
```

## Aberlour (RR)

The route-reflector configuration is very simple. It is configured with IS-IS on its connection to blair and BGP sessions to its route-reflector clients (group rr-clients), as well as a BGP session to the other route-reflector (laphroaig). The configuration cited

here provides a redundant mechanism for BGP route distribution. For more redundancy, a second connection could be added to each route-reflector, but the dual route-reflector design does remove an important single point of failure.

### Junos Style

```
system {
    host-name aberlour;
    root-authentication {
        encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86YhU7CpvPe0"; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2005;
            class super-user;
            authentication {
                encrypted-password "$1$2OY5E3.P$KCIi/yaB9hJ4Z4xy1XIAS1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/0 {
        unit 0 {
            description "Connection to blair ge-0/0/0";
            family inet {
                address 192.168.86.54/30;
            }
            family iso;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.10/32;
            }
            family iso {
                address 49.0002.0102.0008.6010.00;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.10;
    autonomous-system 1;
```

```
}
protocols {
    bgp {
        group rr-clients {
            type internal;
            local-address 10.200.86.10;
            cluster 10.200.86.10;
            neighbor 10.200.86.3;
            neighbor 10.200.86.5;
            neighbor 10.200.86.7;
            neighbor 10.200.86.9;
        }
        group ibgp {
            type internal;
            local-address 10.200.86.10;
            neighbor 10.200.86.11;
        }
    }
    isis {
        level 1 disable;
        level 2 wide-metrics-only;
        interface ge-0/0/0 {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
        }
    }
}
```

Set Style

```
set system host-name aberlour
set system root-authentication encrypted-password "$1$O4d8X3vs$WPM4Gpe6Fy86Y
hU7CpvPe0"
set system login user ps uid 2005
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$20Y5E3.
P$KCIi/yaB9hJ4Z4xy1XIAS1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/0.0 unit 0 description "Connection to blair ge-0/0/0"
set interfaces ge-0/0/0.0 unit 0 family inet address 192.168.86.54/30
set interfaces ge-0/0/0.0 unit 0 family iso
set interfaces lo0 unit 0 family inet address 10.200.86.10/32
set interfaces lo0 unit 0 family iso address 49.0002.0102.0008.6010.00
set routing-options router-id 10.200.86.10
set routing-options autonomous-system 1
set protocols bgp group rr-clients type internal
set protocols bgp group rr-clients local-address 10.200.86.10
set protocols bgp group rr-clients cluster 10.200.86.10
set protocols bgp group rr-clients neighbor 10.200.86.3
set protocols bgp group rr-clients neighbor 10.200.86.5
set protocols bgp group rr-clients neighbor 10.200.86.7
set protocols bgp group rr-clients neighbor 10.200.86.9
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.10
set protocols bgp group ibgp neighbor 10.200.86.11
```

```
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface ge-0/0/0.0 level 1 disable
set protocols isis interface lo0.0 level 1 disable
```

## A Complex Network

- IGP: OSPF
- IGP Scaling: Single area
- Label distribution protocol: Hierarchical LSPs
- RSVP Scaling: Refresh-reductionBGP: Route-reflection
- Growth expectation: Moderate growth
- Additional requirements: Fast resiliency, efficient use of bandwidth, support of MPLS VPN services

Our network now has additional service and network requirements including improved resiliency, edge MPLS VPN services (Layer-3 VPN and VPLS to be exact), and there is a push to make the most efficient use of bandwidth. There is expected to be lots of growth in the edge layer of the network, so route-reflection and hierarchical LSPs are used as scaling mechanisms. The RSVP TE LSPs allow for the use of link-protection for resiliency on the core links, which are far more susceptible to failures than the intra-site links. These RSVP LSPs are configured with the Junos `most-filled` keyword, making for efficient use of the network links, as shown in Figure 4.4.

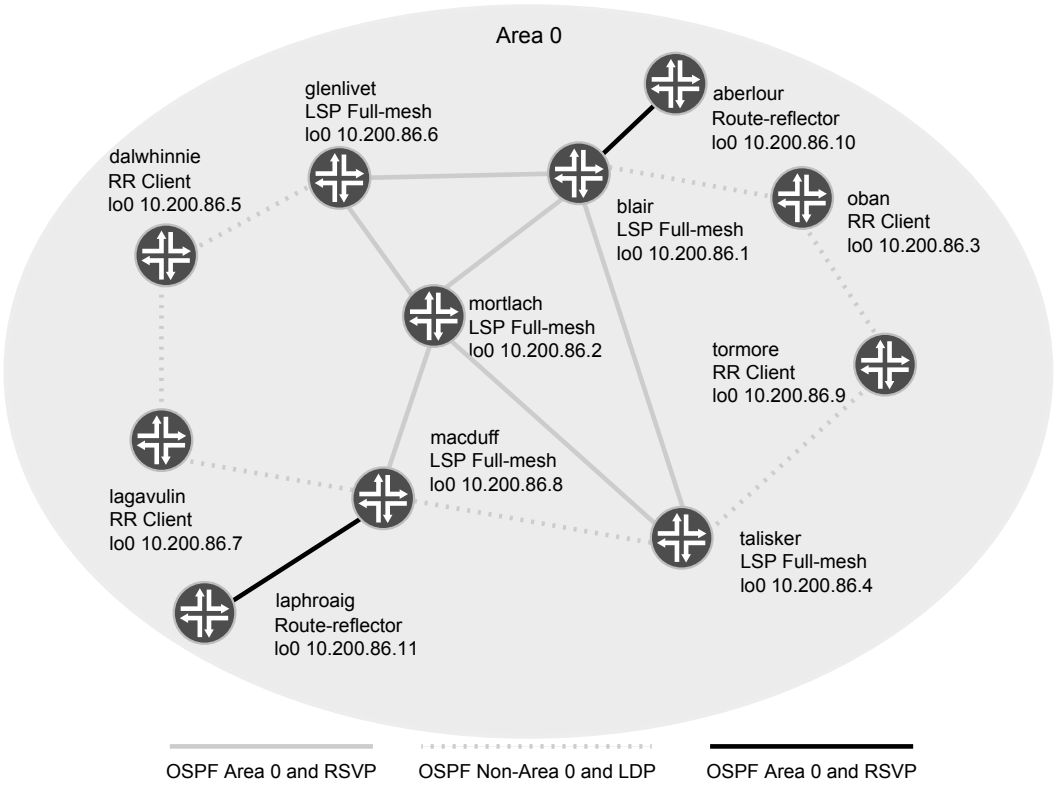



Figure 4.4 The Complex Network

Because this network uses route-reflection, the route-reflectors need to be able to resolve PE next-hop addresses and it is not desirable to extend LSPs to the route-reflectors, so the Junos `routing-options resolution` configuration knob is used to validate next-hop information. To show how the network was built, configurations from dalwhinnie, glenlivet, talisker, tormore, and aberlour are repeated here, which should shows the tunneling for the LSP from dalwhinnie to tormore, and which happens to be taking the dalwhinie-glenlivet-mortlach-talisker-tormore path, although the mortlach hop is hidden from dalwhinnie by the glenlivet to talisker hierarchical LSP. This LSP is tunneled through the core LSP from glenlivet to talisker, which is shown in the configuration.

## Dalwhinnie (PE)

Dalwhinnie is configured with RSVP on the uplink to glenlivet and the cross-connect to lagavulin. These links are also running in OSPF area 0. It is a route-reflector client of the two route-reflectors. All four PE routers follow this design. Note that no changes have to be made on dalwhinnie as a result of the hierarchical LSPs in use, and the PE routers are blind to this scaling technique. Their LSPs are configured with `most-fill`, which encourages efficient use of bandwidth, `link-protection` for fast resiliency, and `auto-bandwidth` to allow for dynamically sized LSPs. Refresh-reduction has been configured under the RSVP interface hierarchies to reduce the overhead of RSVP control messages. BGP is configured with three different address families to support unicast IP routes as well as Layer-2 and Layer-3 VPNs.

### Junos Style

```
system {
   host-name dalwhinnie;
   root-authentication {
      encrypted-password "$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2002;
         class super-user;
         authentication {
            encrypted-password "$1$Vlgdcorz$TRO4TJKkijW62WJtol2EP1"; ##
SECRET-DATA
         }
      }
   }
   services {
      ssh;
   }
   syslog {
      user * {
         any emergency;
      }
      file messages {
         any notice;
         authorization info;
      }
      file interactive-commands {
         interactive-commands any;
      }
   }
}
```

```
interfaces {
    ge-0/0/2 {
        unit 0 {
            description "Connection to glenlivet ge-0/0/2";
                family inet {
                address 192.168.86.6/30;
            }
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to lagavulin ge-0/0/3";
            family inet {
                address 192.168.86.30/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.5/32;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.5;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface ge-0/0/2.0 {
            aggregate;
            link-protection;
        }
        interface ge-0/0/3.0 {
            aggregate;
            link-protection;
        }
    }
    mpls {
        statistics {
            file mpls-stats.txt;
            auto-bandwidth;
        }
        label-switched-path dalwhinnie-to-oban {
            to 10.200.86.3;
            most-fill;
            link-protection;
            auto-bandwidth;
        }
        label-switched-path dalwhinnie-to-lagavulin {
            to 10.200.86.7;
            most-fill;
            link-protection;
            auto-bandwidth;
        }
        label-switched-path dalwhinnie-to-tormore {
            to 10.200.86.9;
            most-fill;
            link-protection;
```

```
            auto-bandwidth;
        }
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.5;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 10.200.86.10;
            neighbor 10.200.86.11;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0;
            interface ge-0/0/2.0;
            interface ge-0/0/3.0;
        }
    }
}
```

### Set Style

```
set system host-name dalwhinnie
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2002
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$Vlgdcorz$TRO4
TJKkijW62WJtol2EP1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/2/0 unit 0 description "Connection to glenlivet ge-0/2/0"
set interfaces ge-0/2/0 unit 0 family inet address 192.168.86.6/30
set interfaces ge-0/2/0 unit 0 family mpls
set interfaces ge-0/3/0 unit 0 description "Connection to lagavulin ge-0/3/0"
set interfaces ge-0/3/0 unit 0 family inet address 192.168.86.30/30
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.5/32
set routing-options router-id 10.200.86.5
set routing-options autonomous-system 1
set protocols rsvp interface ge-0/0/2.0 aggregate
set protocols rsvp interface ge-0/0/2.0 link-protection
set protocols rsvp interface ge-0/0/3.0 aggregate
set protocols rsvp interface ge-0/0/3.0 link-protection
set protocols mpls statistics file mpls-stats.txt
set protocols mpls label-switched-path dalwhinnie-to-oban to 10.200.86.3
set protocols mpls label-switched-path dalwhinnie-to-oban most-fill
```

```
set protocols mpls label-switched-path dalwhinnie-to-oban link-protection
set protocols mpls label-switched-path dalwhinnie-to-oban auto-bandwidth
set protocols mpls label-switched-path dalwhinnie-to-lagavulin to
10.200.86.7
set protocols mpls label-switched-path dalwhinnie-to-lagavulin most-fill
set protocols mpls label-switched-path dalwhinnie-to-lagavulin link-
protection
set protocols mpls label-switched-path dalwhinnie-to-lagavulin auto-
bandwidth
set protocols mpls label-switched-path dalwhinnie-to-tormore to 10.200.86.9
set protocols mpls label-switched-path dalwhinnie-to-tormore most-fill
set protocols mpls label-switched-path dalwhinnie-to-tormore link-
protection
set protocols mpls label-switched-path dalwhinnie-to-tormore auto-bandwidth
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.5
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.10
set protocols bgp group ibgp neighbor 10.200.86.11
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0
```

## Glenlivet (P)

Again, the interesting bits start on the P routers. They utilize a fairly standard OSPF configuration. Each LSP has been configured with `link-protection` and `most-fill`. The link-management piece is the portion that allows for hierarchical LSPs. The `link-management` portion of the protocols hierarchy contains the hierarchical LSP configuration. The `te-link` and `peer` configurations in this stanza create the logical link and interface constructs used in the rest of the configuration.

### Junos Style

```
system {
   host-name glenlivet;
   root-authentication {
      encrypted-password "$1$RJD1JaT1$.SGbFfpwPCS3yc5gqy3Qw."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2000;
         class super-user;
         authentication {
            encrypted-password "$1$VUMxdqLz$hRi6WeofV5MNmnOvKESmV."; ##
SECRET-DATA
         }
      }
   }
   services {
      ssh;
   }
   syslog {
      user * {
         any emergency;
```

```
        }
        file messages {
            any notice;
            authorization info;
        }
    }
}
interfaces {
    ge-0/0/1 {
        unit 0 {
            description "Connection to blair ge-0/0/1";
            family inet {
                address 192.168.86.10/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/2 {
        unit 0 {
            description "Connection to dalwhinnie ge-0/0/2";
            family inet {
                address 192.168.86.5/30;
            }
            family iso;
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            description "Connection to mortlach fe-0/0/1";
            family inet {
                address 192.168.86.46/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.7/32;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.7;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface ge-0/0/1.0 {
            aggregate;
            link-protection;
        }
        interface ge-0/0/2.0 {
            aggregate;
            link-protection;
            aggregate;
        }
        interface ge-0/0/3.0 {
            link-protection;
```

```
    }
    peer-interface peer-talisker;
    peer-interface peer-mortlach;
    peer-interface peer-macduff;
    peer-interface peer-blair;
}
mpls {
    statistics {
        file mpls-stats.txt;
    }
    label-switched-path glenlivet-to-talisker {
        to 10.200.86.4;
        most-fill;
        link-protection;
    }
    label-switched-path glenlivet-to-mortlach {
        to 10.200.86.2;
        most-fill;
        link-protection;
    }
    label-switched-path glenlivet-to-blair {
        to 10.200.86.1;
        most-fill;
        link-protection;
    }
    label-switched-path glenlivet-to-macduff {
        to 10.200.86.8;
        most-fill;
        link-protection;
    }
    interface ge-0/0/1.0;
    interface ge-0/0/2.0;
    interface ge-0/0/2.0;

}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
        peer-interface peer-talisker;
        peer-interface peer-mortlach;
        peer-interface peer-blair;
        peer-interface peer-macduff;
    }
}
link-management {
    te-link glenlivet-to-talisker-te {
        local-address 192.168.87.2;
        remote-address 192.168.87.1;
        label-switched-path glenlivet-to-talisker;
    }
    te-link glenlivet-to-mortlach-te {
        local-address 192.168.87.5;
        remote-address 192.168.87.6;
        te-metric 1;
        label-switched-path glenlivet-to-mortlach;
    }
    te-link glenlivet-to-blair-te {
```

```
            local-address 192.168.87.9;
            remote-address 192.168.87.10;
            te-metric 1;
            label-switched-path glenlivet-to-blair;
        }
        te-link glenlivet-to-macduff-te {
            local-address 192.168.87.13;
            remote-address 192.168.87.14;
            te-metric 1;
            label-switched-path glenlivet-to-macduff;
        }
        peer peer-talisker {
            address 10.200.86.4;
            te-link glenlivet-to-talisker-te;
        }
        peer peer-mortlach {
            address 10.200.86.2;
            te-link glenlivet-to-mortlach-te;
        }
        peer peer-blair {
            address 10.200.86.1;
            te-link glenlivet-to-blair-te;
        }
        peer peer-macduff {
            address 10.200.86.8;
            te-link glenlivet-to-macduff-te;
        }
    }
}
```

### Set Style

```
set system host-name glenlivet
set system root-authentication encrypted-password "$1$RJD1JaT1$.
SGbFfpwPCS3yc5gqy3Qw."
set system login user ps uid 2000
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$VUMxdqLz$hRi6
WeofV5MNmnOvKESmV."
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set interfaces ge-0/0/1 unit 0 description "Connection to blair ge-0/0/1"
set interfaces ge-0/0/1 unit 0 family inet address 192.168.86.10/30
set interfaces ge-0/0/1 unit 0 family iso
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces ge-0/0/2 unit 0 description "Connection to dalwhinnie ge-
0/0/2"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.5/30
set interfaces ge-0/0/2 unit 0 family iso
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 description "Connection to mortlach fe-0/0/1"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.46/30
set interfaces ge-0/0/3 unit 0 family iso
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.7/32
set routing-options router-id 10.200.86.7
set routing-options autonomous-system 1
set protocols rsvp interface ge-0/0/1.0 aggregate
set protocols rsvp interface ge-0/0/1.0 link-protection
```

```
set protocols rsvp interface ge-0/0/2.0 aggregate
set protocols rsvp interface ge-0/0/2.0 link-protection
set protocols rsvp interface ge-0/0/3.0 aggregate
set protocols rsvp interface ge-0/0/3.0 link-protection
set protocols rsvp peer-interface peer-talisker
set protocols rsvp peer-interface peer-mortlach
set protocols rsvp peer-interface peer-macduff
set protocols rsvp peer-interface peer-blair
set protocols mpls label-switched-path glenlivet-to-talisker to 10.200.86.4
set protocols mpls label-switched-path glenlivet-to-talisker most-fill
set protocols mpls label-switched-path glenlivet-to-talisker link-
protection
set protocols mpls label-switched-path glenlivet-to-mortlach to 10.200.86.2
set protocols mpls label-switched-path glenlivet-to-mortlach most-fill
set protocols mpls label-switched-path glenlivet-to-mortlach link-
protection
set protocols mpls label-switched-path glenlivet-to-blair to 10.200.86.1
set protocols mpls label-switched-path glenlivet-to-blair most-fill
set protocols mpls label-switched-path glenlivet-to-blair link-protection
set protocols mpls label-switched-path glenlivet-to-macduff to 10.200.86.8
set protocols mpls label-switched-path glenlivet-to-macduff most-fill
set protocols mpls label-switched-path glenlivet-to-macduff link-protection
set protocols mpls interface ge-0/0/1.0
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0
set protocols ospf area 0.0.0.0 peer-interface peer-talisker
set protocols ospf area 0.0.0.0 peer-interface peer-mortlach
set protocols ospf area 0.0.0.0 peer-interface peer-blair
set protocols ospf area 0.0.0.0 peer-interface peer-macduff
set protocols link-management te-link glenlivet-to-talisker-te local-
address 192.168.87.2
set protocols link-management te-link glenlivet-to-talisker-te remote-
address 192.168.87.1
set protocols link-management te-link glenlivet-to-talisker-te label-
switched-path glenlivet-to-talisker
set protocols link-management te-link glenlivet-to-mortlach-te local-
address 192.168.87.5
set protocols link-management te-link glenlivet-to-mortlach-te remote-
address 192.168.87.6
set protocols link-management te-link glenlivet-to-mortlach-te te-metric 1
set protocols link-management te-link glenlivet-to-mortlach-te label-
switched-path glenlivet-to-mortlach
set protocols link-management te-link glenlivet-to-blair-te local-address
192.168.87.9
set protocols link-management te-link glenlivet-to-blair-te remote-address
192.168.87.10
set protocols link-management te-link glenlivet-to-blair-te te-metric 1
set protocols link-management te-link glenlivet-to-blair-te label-switched-
path glenlivet-to-blair
set protocols link-management te-link glenlivet-to-macduff-te local-address
192.168.87.13
set protocols link-management te-link glenlivet-to-macduff-te remote-
address 192.168.87.14
set protocols link-management te-link glenlivet-to-macduff-te te-metric 1
set protocols link-management te-link glenlivet-to-macduff-te label-
switched-path glenlivet-to-macduff
set protocols link-management peer peer-talisker address 10.200.86.4
set protocols link-management peer peer-talisker te-link glenlivet-to-
```

```
talisker-te
set protocols link-management peer peer-mortlach address 10.200.86.2
set protocols link-management peer peer-mortlach te-link glenlivet-to-
mortlach-te
set protocols link-management peer peer-blair address 10.200.86.1
set protocols link-management peer peer-blair te-link glenlivet-to-blair-te
set protocols link-management peer peer-macduff address 10.200.86.8
set protocols link-management peer peer-macduff te-link glenlivet-to-
macduff-te
```

## Talisker (P)

Talisker's P configuration looks very similar to glenlivet's.

### Junos Style

```
system {
   host-name talisker;
   root-authentication {
      encrypted-password "$1$RJD1JaT1$.SGbFfpwPCS3yc5gqy3Qw."; ## SECRET-
DATA
   }
   login {
      user ps {
         uid 2000;
         class super-user;
         authentication {
            encrypted-password "$1$grlc52X7$RF98wUcuOF.b2FiZZKKLC1"; ##
SECRET-DATA
         }
      }
   }
   services {
      ssh;
   }
   syslog {
      user * {
         any emergency;
      }
      file messages {
         any notice;
         authorization info;
      }
   }
}
interfaces {
   fe-0/0/1 {
      description "Connection to macduff ge-0/0/1";
      unit 0 {
         family inet {
            address 192.168.86.13/30;
         }
         family mpls;
      }
   }
   e1-1/0/0 {
      description "Connection to mortlach e1-0/0/0";
      unit 0 {
         family inet {
            address 192.168.86.21/30;
         }
         family mpls;
```

```
        }
    }
    fe-2/0/0 {
        description "Connection to blair ge-0/0/3";
        unit 0 {
            family inet {
                address 192.168.86.17/30;
            }
            family mpls;
        }
    }
    fe-2/0/1 {
        description "Connection to tormore ge-0/0/2";
        unit 0 {
            family inet {
                address 192.168.86.34/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.4/32;
            }
        }
    }
}
routing-options {
    router-id 10.200.86.4;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface fe-0/0/1.0 {
            aggregate;
            link-protection;
        }
        interface e1-1/0/0.0 {
            aggregate;
            link-protection;
        }
        interface fe-2/0/0.0 {
            aggregate;
            link-protection;
        }
        interface fe-2/0/1.0 {
            aggregate;
            link-protection;
        }
        peer-interface peer-glenlivet;
        peer-interface peer-mortlach;
        peer-interface peer-blair;
        peer-interface peer-macduff;
    }
    mpls {
        label-switched-path talisker-to-glenlivet {
            to 10.200.86.7;
            most-fill;
            link-protection;
        }
        label-switched-path talisker-to-mortlach {
            to 10.200.86.2;
```

```
            most-fill;
            link-protection;
        }
        label-switched-path talisker-to-blair {
            to 10.200.86.1;
            most-fill;
            link-protection;
        }
        label-switched-path talisker-to-macduff {
            to 10.200.86.8;
            most-fill;
            link-protection;
        }
        interface fe-0/0/1.0;
        interface e1-1/0/0.0;
        interface fe-2/0/0.0;
        interface fe-2/0/1.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
interface fe-0/0/1.0;
interface e1-1/0/0.0;
interface fe-2/0/0.0;
interface fe-2/0/1.0;
            peer-interface peer-glenlivet;
            peer-interface peer-mortlach;
            peer-interface peer-blair;
            peer-interface peer-macduff;
        }
    }
    link-management {
        te-link talisker-to-glenlivet-te {
            local-address 192.168.87.1;
            remote-address 192.168.87.2;
            te-metric 1;
            label-switched-path talisker-to-glenlivet;
        }
        te-link talisker-to-mortlach-te {
            local-address 192.168.87.5;
            remote-address 192.168.87.6;
            te-metric 1;
            label-switched-path talisker-to-mortlach;
        }
        te-link talisker-to-blair-te {
            local-address 192.168.87.9;
            remote-address 192.168.87.10;
            te-metric 1;
            label-switched-path talisker-to-blair;
        }
        te-link talisker-to-macduff-te {
            local-address 192.168.87.13;
            remote-address 192.168.87.14;
            te-metric 1;
            label-switched-path talisker-to-macduff;
        }
        peer peer-glenlivet {
            address 10.200.86.7;
            te-link talisker-to-glenlivet-te;
        }
```

```
        peer peer-mortlach {
            address 10.200.86.2;
            te-link talisker-to-mortlach-te;
        }
        peer peer-blair {
            address 10.200.86.1;
            te-link talisker-to-blair-te;
        }
        peer peer-macduff {
            address 10.200.86.8;
            te-link talisker-to-macduff-te;
        }
    }
}
```

### Set Style

```
set system host-name talisker
set system root-authentication encrypted-password "$1$RJD1JaT1$.
SGbFfpwPCS3yc5gqy3Qw."
set system login user ps uid 2000
set system login user ps class super-user
set system login user ps authentication encrypted-password
"$1$grlc52X7$RF98wUcuOF.b2FiZZKKLC1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set interfaces fe-0/0/1 description "Connection to macduff ge-0/0/1"
set interfaces fe-0/0/1 unit 0 family inet address 192.168.86.13/30
set interfaces fe-0/0/1 unit 0 family mpls
set interfaces e1-1/0/0 description "Connection to mortlach e1-0/0/0"
set interfaces e1-1/0/0 unit 0 family inet address 192.168.86.21/30
set interfaces e1-1/0/0 unit 0 family mpls
set interfaces fe-2/0/0 description "Connection to blair ge-0/0/3"
set interfaces fe-2/0/0 unit 0 family inet address 192.168.86.17/30
set interfaces fe-2/0/0 unit 0 family mpls
set interfaces fe-2/0/1 description "Connection to tormore ge-0/0/2"
set interfaces fe-2/0/1 unit 0 family inet address 192.168.86.34/30
set interfaces fe-2/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.4/32
set routing-options router-id 10.200.86.4
set routing-options autonomous-system 1
set protocols rsvp interface fe-0/0/1.0 aggregate
set protocols rsvp interface fe-0/0/1.0 link-protection
set protocols rsvp interface e1-1/0/0.0 aggregate
set protocols rsvp interface e1-1/0/0.0 link-protection
set protocols rsvp interface fe-2/0/0.0 aggregate
set protocols rsvp interface fe-2/0/0.0 link-protection
set protocols rsvp interface fe-2/0/1.0 aggregate
set protocols rsvp interface fe-2/0/1.0 link-protection
set protocols rsvp peer-interface peer-glenlivet
set protocols rsvp peer-interface peer-mortlach
set protocols rsvp peer-interface peer-blair
set protocols rsvp peer-interface peer-macduff
set protocols mpls label-switched-path talisker-to-glenlivet to 10.200.86.7
set protocols mpls label-switched-path talisker-to-glenlivet most-fill
set protocols mpls label-switched-path talisker-to-glenlivet link-
protection
set protocols mpls label-switched-path talisker-to-mortlach to 10.200.86.2
```

```
set protocols mpls label-switched-path talisker-to-mortlach most-fill
set protocols mpls label-switched-path talisker-to-mortlach link-protection
set protocols mpls label-switched-path talisker-to-blair to 10.200.86.1
set protocols mpls label-switched-path talisker-to-blair most-fill
set protocols mpls label-switched-path talisker-to-blair link-protection
set protocols mpls label-switched-path talisker-to-macduff to 10.200.86.8
set protocols mpls label-switched-path talisker-to-macduff most-fill
set protocols mpls label-switched-path talisker-to-macduff link-protection
set protocols mpls interface fe-0/0/1.0
set protocols mpls interface e1-1/0/0.0
set protocols mpls interface fe-2/0/0.0
set protocols mpls interface fe-2/0/1.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-0/0/1.0
set protocols ospf area 0.0.0.0 interface e1-1/0/0.0
set protocols ospf area 0.0.0.0 interface fe-2/0/0.0
set protocols ospf area 0.0.0.0 interface fe-2/0/1.0
set protocols ospf area 0.0.0.0 peer-interface peer-glenlivet
set protocols ospf area 0.0.0.0 peer-interface peer-mortlach
set protocols ospf area 0.0.0.0 peer-interface peer-blair
set protocols ospf area 0.0.0.0 peer-interface peer-macduff
set protocols link-management te-link talisker-to-glenlivet-te local-address
192.168.87.1
set protocols link-management te-link talisker-to-glenlivet-te remote-
address 192.168.87.2
set protocols link-management te-link talisker-to-glenlivet-te te-metric 1
set protocols link-management te-link talisker-to-glenlivet-te label-
switched-path talisker-to-glenlivet
set protocols link-management te-link talisker-to-mortlach-te local-address
192.168.87.5
set protocols link-management te-link talisker-to-mortlach-te remote-address
192.168.87.6
set protocols link-management te-link talisker-to-mortlach-te te-metric 1
set protocols link-management te-link talisker-to-mortlach-te label-
switched-path talisker-to-mortlach
set protocols link-management te-link talisker-to-blair-te local-address
192.168.87.9
set protocols link-management te-link talisker-to-blair-te remote-address
192.168.87.10
set protocols link-management te-link talisker-to-blair-te te-metric 1
set protocols link-management te-link talisker-to-blair-te label-switched-
path talisker-to-blair
set protocols link-management te-link talisker-to-macduff-te local-address
192.168.87.13
set protocols link-management te-link talisker-to-macduff-te remote-address
192.168.87.14
set protocols link-management te-link talisker-to-macduff-te te-metric 1
set protocols link-management te-link talisker-to-macduff-te label-switched-
path talisker-to-macduff
set protocols link-management peer peer-glenlivet address 10.200.86.7
set protocols link-management peer peer-glenlivet te-link talisker-to-
glenlivet-te
set protocols link-management peer peer-mortlach address 10.200.86.2
set protocols link-management peer peer-mortlach te-link talisker-to-
mortlach-te
set protocols link-management peer peer-blair address 10.200.86.1
set protocols link-management peer peer-blair te-link talisker-to-blair-te
set protocols link-management peer peer-macduff address 10.200.86.8
set protocols link-management peer peer-macduff te-link talisker-to-macduff-
te
```

## Tormore (PE)

Tormore's configuration is included for completeness, but closely matches dalwhinnie's configuration.

### Junos Style

```
system {
    host-name tormore;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2002;
            class super-user;
            authentication {
                encrypted-password "$1$Vlgdcorz$TRO4TJKkijW62WJtol2EP1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/2 {
        description "Connection to talisker ge-2/0/1";
        unit 0 {
            family inet {
                address 192.168.86.33/30;
            }
            family mpls;
        }
    }
    ge-0/0/3 {
        description "Connection to oban ge-0/0/3";
        unit 0 {
            family inet {
                address 192.168.86.37/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.9/32;
```

```
            }
        }
    }
}
routing-options {
    router-id 10.200.86.9;
    autonomous-system 1;
}
protocols {
    rsvp {
        interface ge-0/0/2.0 {
            aggregate;
            link-protection;
        }
        interface ge-0/0/3.0 {
            aggregate;
            link-protection;
        }
    }
    mpls {
        statistics {
            file mpls-stats.txt;
        }
        label-switched-path tormore-to-oban {
            to 10.200.86.3;
            most-fill;
            link-protection;
            auto-bandwidth;
        }
        label-switched-path tormore-to-dalwhinnie {
            to 10.200.86.5;
            most-fill;
            link-protection;
            auto-bandwidth;
        }
        label-switched-path tormore-to-lagavulin {
            to 10.200.86.7;
            most-fill;
            link-protection;
            auto-bandwidth;
        }
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.200.86.9;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 10.200.86.10;
            neighbor 10.200.86.11;
        }
    }
    ospf {
        traffic-engineering;
```

```
        area 0.0.0.0 {
            interface lo0.0;
            interface ge-0/0/2.0;
            interface ge-0/0/3.0;
        }
    }
}
```

### Set Style

```
set system host-name tormore
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2002
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$Vlgdcorz$TRO
4TJKkijW62WJtol2EP1"
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/2 description "Connection to talisker fe-2/0/1"
set interfaces ge-0/0/2 unit 0 family inet address 192.168.86.33/30
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 description "Connection to oban ge-0/0/3"
set interfaces ge-0/0/3 unit 0 family inet address 192.168.86.37/30
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.200.86.9/32
set routing-options router-id 10.200.86.9
set routing-options autonomous-system 1
set protocols rsvp interface ge-0/0/2.0 aggregate
set protocols rsvp interface ge-0/0/2.0 link-protection
set protocols rsvp interface ge-0/0/3.0 aggregate
set protocols rsvp interface ge-0/0/3.0 link-protection
set protocols mpls statistics file mpls-stats.txt
set protocols mpls label-switched-path tormore-to-oban to 10.200.86.3
set protocols mpls label-switched-path tormore-to-oban most-fill
set protocols mpls label-switched-path tormore-to-oban link-protection
set protocols mpls label-switched-path tormore-to-oban auto-bandwidth
set protocols mpls label-switched-path tormore-to-dalwhinnie to 10.200.86.5
set protocols mpls label-switched-path tormore-to-dalwhinnie most-fill
set protocols mpls label-switched-path tormore-to-dalwhinnie link-
protection
set protocols mpls label-switched-path tormore-to-dalwhinnie auto-bandwidth
set protocols mpls label-switched-path tormore-to-lagavulin to 10.200.86.7
set protocols mpls label-switched-path tormore-to-lagavulin most-fill
set protocols mpls label-switched-path tormore-to-lagavulin link-protection
set protocols mpls label-switched-path tormore-to-lagavulin auto-bandwidth
set protocols mpls interface ge-0/0/2.0
set protocols mpls interface ge-0/0/3.0
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.9
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.10
set protocols bgp group ibgp neighbor 10.200.86.11
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0
```

## Aberlour (RR)

The route-reflector configuration is very simple. It is configured with OSPF on its connection to blair and BGP sessions to its route-reflector clients (group rr-clients) as well as a BGP session to the other route-reflector (laphroaig). This configuration provides a redundant mechanism for BGP route distribution. For more redundancy, a second connection could be added to each route-reflector, but the dual route-reflector design does limit an important single point of failure. Aberlour is also configured with the Junos `routing-options resolution` keywords to allow VPN routes to be validated even though it does not have an MPLS reachability to the PE routers.

### Junos Style

```
system {
    host-name aberlour;
    root-authentication {
        encrypted-password "$1$yCV3hpZD$EWpPM8TW8exo0r/6v.Yfk."; ## SECRET-
DATA
    }
    login {
        user ps {
            uid 2002;
            class super-user;
            authentication {
                encrypted-password "$1$Vlgdcorz$TRO4TJKkijW62WJtol2EP1"; ##
SECRET-DATA
            }
        }
    }
    services {
        ssh;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/0 {
        description "Connection to blair ge-0/0/0";
        unit 0 {
            family inet {
                address 192.168.86.54/30;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.200.86.10/32;
            }
        }
    }
```

```
}
routing-options {
    router-id 10.200.86.10;
    autonomous-system 1;
    resolution {
        rib bgp.l3vpn.0 {
            resolution-ribs inet.0;
        }
    }
}
protocols {
    bgp {
        group rr-clients {
            type internal;
            local-address 10.200.86.10;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            cluster 10.200.86.10;
            neighbor 10.200.86.5;
            neighbor 10.200.86.7;
            neighbor 10.200.86.3;
            neighbor 10.200.86.9;
        }
        group ibgp {
            type internal;
            local-address 10.200.86.10;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 10.200.86.11;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface ge-0/0/0.0;
            interface lo0.0;
        }
    }
}
```

### Set Style

```
set system host-name aberlour
set system root-authentication encrypted-password
"$1$yCV3hpZD$EWpPM8TW8exoOr/6v.Yfk."
set system login user ps uid 2002
set system login user ps class super-user
set system login user ps authentication encrypted-password "$1$Vlgdcorz$TRO
4TJKkijW62WJtol2EP1"
```

```
set system services ssh
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set interfaces ge-0/0/0 description "Connection to blair ge-0/0/0"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.86.54/30
set interfaces lo0 unit 0 family inet address 10.200.86.10/32
set routing-options router-id 10.200.86.10
set routing-options autonomous-system 1
set routing-options resolution rib bgp.l3vpn.0 resolution-ribs inet.0
set protocols bgp group rr-clients type internal
set protocols bgp group rr-clients local-address 10.200.86.10
set protocols bgp group rr-clients family inet unicast
set protocols bgp group rr-clients family inet-vpn unicast
set protocols bgp group rr-clients family l2vpn signaling
set protocols bgp group rr-clients cluster 10.200.86.10
set protocols bgp group rr-clients neighbor 10.200.86.5
set protocols bgp group rr-clients neighbor 10.200.86.7
set protocols bgp group rr-clients neighbor 10.200.86.3
set protocols bgp group rr-clients neighbor 10.200.86.9
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.200.86.10
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 10.200.86.11
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0
```

## Summary

There are many different ways to solve network problems, but it can be difficult to figure out which way is the best for you. These examples provide working solutions to some common network problems. It is unlikely that your network's needs line up exactly with any of these example networks, but you can pick and choose from these examples to fit your need. This is one of the most powerful aspects of the modularized and distributed architecture of Junos because it allows the user to pick and choose configuration segments, easily load, activate and disable them, and even run different protocols in parallel for easy conversion.

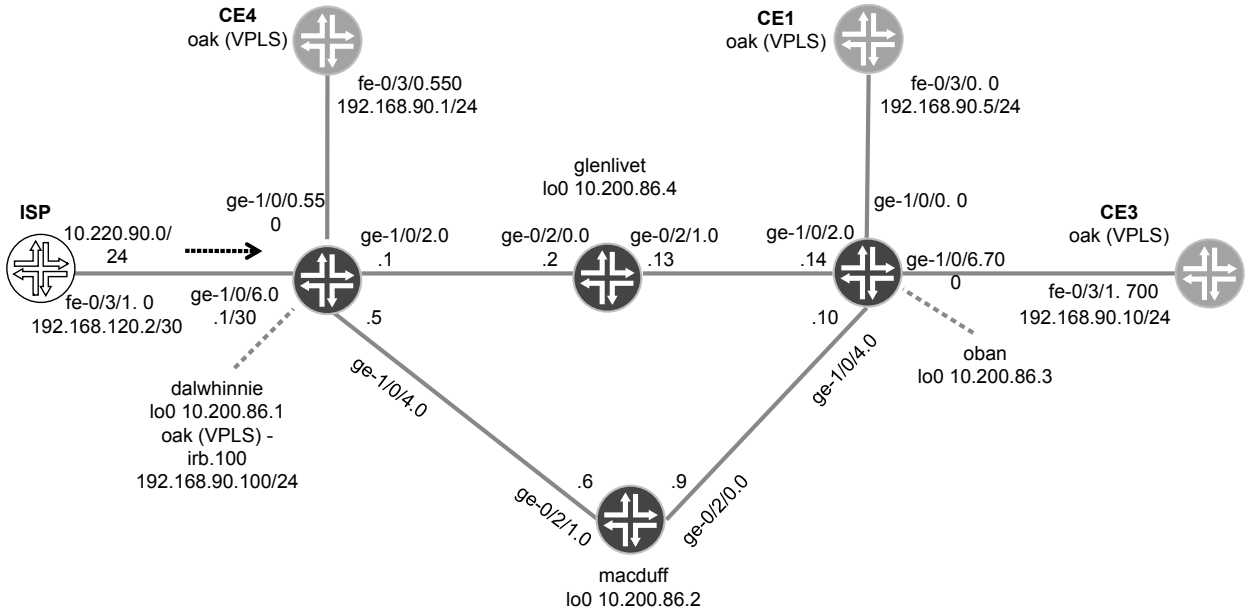
The hope was that these examples would not only provide a direct 'cut-and-paste' style approach to your MPLS deployment, but also bring attention to the different ways in which MPLS and Junos can be used to develop a better-fitting, customized design.

TIP If you visit www.juniper.net/dayone, and then follow the path to this book's download page, you'll find a free *Copy and Paste* edition of this book. Its rich-text format allows the file to be opened in various text editors for direct placement of the Junos configurations.

# Appendices

## Appendix A: VPLS in MX Series Routers

The MX Series of Juniper Networks routers adds additional functionality and ease of configuration to VPLS, and this appendix explores some of those additional features. Figure A.1 shows a new sample network for this appendix, with PE routers dalwhinnie and oban (MX80 routers), along with three CE routers (CE1, CE3, CE4), and two core routers (glenlivet and macduff); refer to this figure in the following case studies of Appendix A.




All core interface IP addresses are in /30 subnets in the 192.168.86/24 network unless otherwise noted.

Figure A.1 Sample Network with MX80 PEs

The VPLS configuration for the CE-facing interface and the VPLS instance *oak* on the dalwhinnie router is right here (reference Chapter 2 if you want to compare the MX80 configurations here to those on the J-series routers):

```
regress@dalwhinnie> show configuration routing-instances oak
instance-type vpls;
vlan-id none;
interface ge-1/0/0.550;
routing-interface irb.100;
route-distinguisher 10.200.86.1:100;
vrf-target target:300:200;
protocols {
    vpls {
        site-range 5;
        no-tunnel-services;
        site ce4 {
            site-identifier 4;
            interface ge-1/0/0.550;
        }
    }
}
regress@dalwhinnie> show configuration interfaces
ge-1/0/0 {
    vlan-tagging;
```

```
    speed 100m;
    encapsulation vlan-vpls;
    unit 550 {
        encapsulation vlan-vpls;
        vlan-id 550;
        family vpls;
    }
}
 . . .
 . . .
irb {
    unit 100 {
        family inet {
            address 192.168.90.100/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 127.0.0.1/32;
            address 10.200.86.1/32;
        }
    }
}
```

The first difference of note on the MX configuration is the lack of *input-* and *output-vlan-maps* on the CE-facing interface on the MX80 PE. Instead, there is a *vlan-id* value specified (a value of *none* in this example) in the VPLS instance. In Chapter 2 the input-vlan-map was used to remove the vlan tag so that the packet could traverse the core with no vlan tag; and the output-vlan-map was used to add the appropriate vlan-tag back to the packet upon egress from the destination PE. This mapping has to be done so that the packet can enter and exit the core with the appropriate vlan-id value and vlan-stack height.

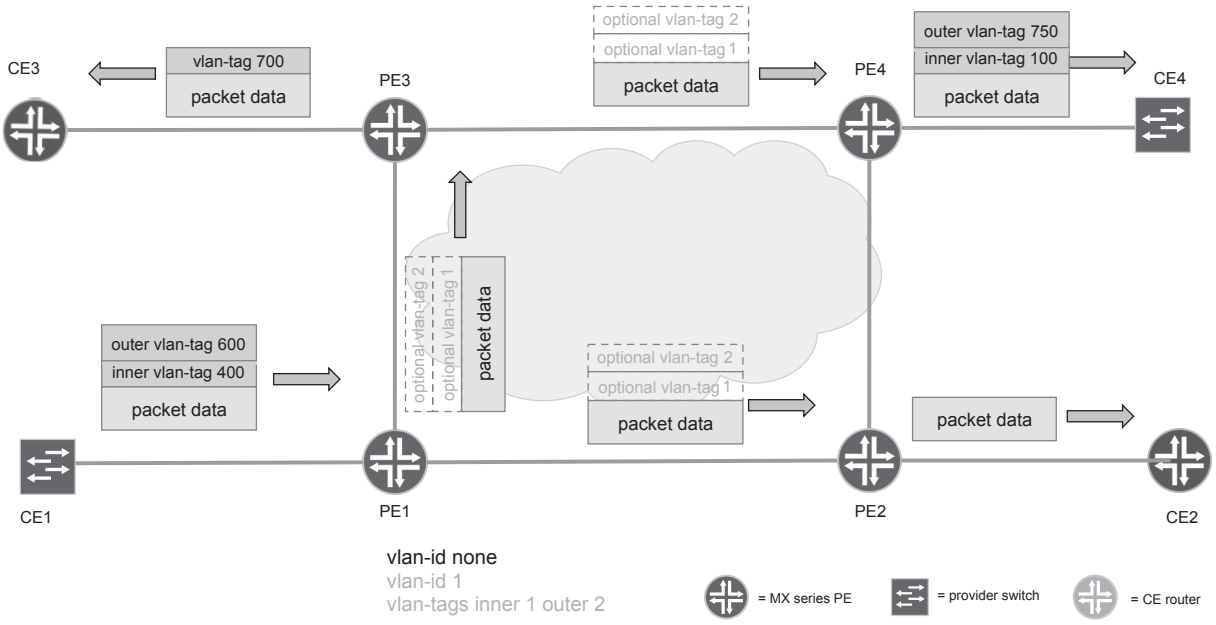
In essence, the vlan-map configurations in Chapter 2 normalized the vlan-stack height to zero for transit across the MPLS core. This normalization is necessary for CEs in a common VPLS instance to speak to each other if some are accessing a PE via an interface that uses vlan-tagging and others are accessing a PE via an interface that does not use vlan-tagging at all. Note that if PEs have different vlan-ids configured on their CE-facing VPLS interfaces, then it is necessary to normalize vlans across the core in order for the CEs to speak to each other. Using the input-vlan-map to normalize the vlan-stack height to zero at the ingress PE for transit across the core ensures that the output-vlan-map places the correct vlan-id value (or none) on the packet as at the egress PE's CE-facing interface.

But the MX Series is capable of accepting packets with zero, or one or two stacked vlan tags upon ingress to the PE *without* using vlan-maps. The *vlan-id* configuration in the VPLS instance automatically normalizes the vlan-stack height for the packets traversing the core. The MX Series routers are able to automatically calculate the required vlan-tagging operation for a given packet's ingress/egress interface.

NOTE In this MX example, the vlan-stack height is normalized to zero (vlan-id none). To normalize to a stack height value of one, the vlan-id value needs to be set to a nonzero value. The *vlan-tags* configuration provides the same functionality as *vlan-id* with the only difference being that *vlan-tags* requires two vlan values (*inner* and *outer*) and so normalizes the vlan-stack height to two.

BEST PRACTICE While it's necessary to normalize to the same vlan-stack height for a common VPLS instance, it is *not* necessarily required that each PE have the *same* vlan values configured in the vlan-stack for the given VPLS instance on an MX Series router. It is, however, a best practice to do so. Additionally, vlan stack values do not have to be unique on a per-VPLS instance basis – two unique VPLS instances can have the same vlan-id/vlan-tags values configured. That's because the vlan-id values in the core are not used to distinguish or separate traffic; they are primarily used to carry 802.1p class-of-service markings.

Figure A.2 below shows a sample VPLS instance, on MX series PEs, normalized to zero vlans across the core (*vlan-id none*). Normalizing to zero vlans across the core results in the lowest per-packet overhead because vlan tags are not transported.




Note: LSP labels not shown.

Figure A.2 Vlan-Normalization Illustration

Figure A.2 illustrates how the vlan-stack height can be normalized to zero, one, or two vlan-tags for transport across the core in VPLS, ensuring that the packet is transmitted out the egress PE interface only with the appropriate vlan-id values and vlan-stack height. Let's trace it while you can follow it on the page. CE1 transmits a packet to PE1 with a vlan-stack height of two. If the vlan-stack is normalized to zero across the core, then PE1 removes both vlan-tags (pop 600 pop 400) from the packet before transport to the remote PEs. PE2 does not add any vlans to the packet to transmit to CE2; PE3 adds only one vlan-tag (push 700); PE4 adds inner tag 100 and outer tag 750 (push 100 push 750). If it is necessary to normalize to one vlan across the core (vlan-id 1 in figure A.2), then PE1 removes outer vlan-tag 600 and swaps vlan-tag 400 for vlan-id 1 (pop 600 swap 1); PE2 removes vlan-id 1 from the packet (pop 1) before transmitting to CE2; PE3 swaps vlan-id 1 for vlan-id 700 before transmitting to CE3 (swap 700); PE4 swaps vlan-id 1 for vlan-id 100 and pushes an additional tag with value 750 (*swap 100 push 750*).

### Try It Yourself: Map Vlan-stack Operations

If the VPLS instance in figure A.2 is configured for vlan-tags inner 1 outer 2, what are the vlan-tag operations for the packet upon entry to PE1? What are the vlan-tag operations upon egress from PE2, PE3, and PE4? Post your answers on the J-Net page for this book at www.juniper.net/dayone.

The example with the oak VPLS instance normalizes to a vlan-stack height of zero. Depending on class-of-service requirements (which are outside the scope of this book, thank you), it may be necessary to normalize vlan-id stack height to a value of one or two for packets transiting the core.

NOTE The VLAN normalization behavior inherent to the MX-Series architecture is sometimes referred to as "implicit VLAN translation" and is sometimes used by Service Providers to achieve complex VLAN translation schemes with very minimal and straightforward configuration.

Another item to note is the optional presence of an *irb* (integrated routing and bridging) interface in the VPLS instance. Having this interface available in the VPLS instance allows for some additional capabilities: it allows the service provider to offer a *gateway in the cloud* for Internet access to the VPLS customer and also allows for additional troubleshooting capability for the ISP. Let's examine these capabilities, as they are interesting.

Placing an irb interface in VPLS can add value when a service that includes VPLS and Internet access over that same circuit is desired because it may not be practical to have a router at one of the end VPLS sites because of the management overhead and the potential redundancy problems that that can entail. For example, if one of the VPLS end sites has an onsite router for Internet access and the router or the site's access circuit goes down, all the other sites in the common VPLS domain lose Internet connectivity. While it's possible to add a redundant router at another CE site in the VPLS instance, doing so introduces more management overhead and potential failover procedures to address.

Having VPLS with Internet access via an irb interface on a PE router provides an Internet gateway *in the cloud* for the VPLS user: any CE site in the instance *oak* could reach external routes via the irb interface, regardless of whether any other CE sites are down (the assumption, of course, is that the PE with the irb interface does not go down or lose external connectivity!). Other similar uses for an irb interface in a VPLS instance include providing VPLS-L3VPN connectivity or using the irb interface to help migrate a customer from VPLS to L3VPN service.

NOTE An important point that needs to be made about a permanent irb interface in a VPLS instance is that the network PE routers become aware of the CE routers' address space, due to the presence of the irb interface Layer 3 address in the master routing instance. *It is necessary, then, for the CE routers to have and use their designated public IP space for their CE addressing scheme.*

Let's look at some sample output. Here oak's address space is shown to be directly reachable, via the irb.100 interface, from dalwhinnie's master instance:

```
regress@dalwhinnie> show route 192.168.90.5
inet.0: 20 destinations, 20 routes (19 active, 0 holddown, 1 hidden)
```

```
+ = Active Route, - = Last Active, * = Both
192.168.90.0/24    *[Direct/0] 3d 00:16:18
                  > via irb.100
regress@dalwhinnie> ping 192.168.90.5
PING 192.168.90.5 (192.168.90.5): 56 data bytes
64 bytes from 192.168.90.5: icmp_seq=0 ttl=64 time=0.984 ms
64 bytes from 192.168.90.5: icmp_seq=1 ttl=64 time=0.933 ms
64 bytes from 192.168.90.5: icmp_seq=2 ttl=64 time=0.899 ms
64 bytes from 192.168.90.5: icmp_seq=3 ttl=64 time=0.931 ms
64 bytes from 192.168.90.5: icmp_seq=4 ttl=64 time=0.905 ms
^C
--- 192.168.90.5 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.899/0.930/0.984/0.030 ms
```

Let's look at router CE4 in the VPLS instance oak. In a typical VPLS service, CE4 would only be able to send traffic to other sites within oak. With the addition of the irb interface, users in oak now have the ability to connect to sites outside the VPLS instance, including some core network internal addresses. Here CE4 is able to ping to 192.168.86.1, ge-1/0/2.0 on dalwhinnie:

```
regress@ce4> ping 192.168.86.1
PING 192.168.86.1 (192.168.86.1): 56 data bytes
64 bytes from 192.168.86.1: icmp_seq=0 ttl=64 time=0.816 ms
64 bytes from 192.168.86.1: icmp_seq=1 ttl=64 time=0.767 ms
64 bytes from 192.168.86.1: icmp_seq=2 ttl=64 time=0.810 ms
64 bytes from 192.168.86.1: icmp_seq=3 ttl=64 time=0.802 ms
^C
--- 192.168.86.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.767/0.799/0.816/0.019 ms
```

It may not make sense to allow VPLS users to send traffic to internal core addresses or other sensitive, restricted sites. You can remedy this potential problem by configuring an input filter on the VPLS instance's irb interface. Here the example applies a filter that denies access to core internal addresses but allows access to any other networks:

```
[edit interfaces irb]
regress@dalwhinnie# show
unit 100 {
    family inet {
        address 192.168.90.100/24;
    }
}
[edit interfaces irb]
regress@dalwhinnie# set unit 100 family inet filter input vpls-internet-
access
[edit interfaces irb]
regress@dalwhinnie# show
unit 100 {
    family inet {
        filter {
            input vpls-internet-access;
        }
        address 192.168.90.100/24;
    }
}
```

So the *vpls-internet-access* filter applied to irb.100 discards any traffic bound for core network addresses, logs the attempt, and allows all other destinations, shown here:

```
regress@dalwhinnie> show configuration firewall family inet filter vpls-
internet-access
term 10 {
    from {
        destination-prefix-list {
            internal-routes;
        }
    }
    then {
        count vpls-internal-attempt;
        discard;
    }
}
term 20 {
    then accept;
}
regress@dalwhinnie> show configuration policy-options
prefix-list internal-routes {
    192.168.86.0/24;
}
 . . .
 . . .
 . . .
regress@dalwhinnie>
```

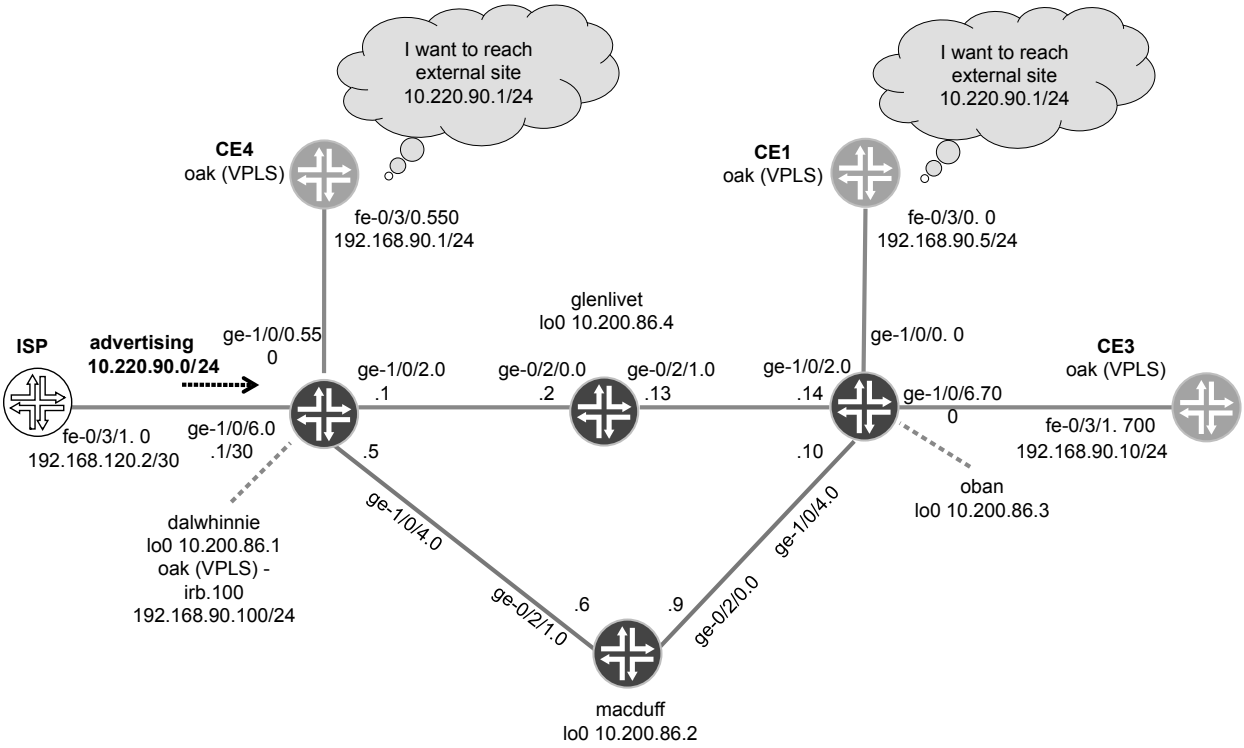
With the application of this filter, CE routers in oak can no longer reach internal core network addresses.

```
regress@ce4> ping 192.168.86.1
PING 192.168.86.1 (192.168.86.1): 56 data bytes
^C
--- 192.168.86.1 ping statistics ---
191 packets transmitted, 0 packets received, 100% packet loss

regress@ce4>
```

As CE4 attempts to ping 192.168.86.1, the counters on the *vpls-internet-access* filter increment, showing that the filter is indeed blocking the traffic:

```
regress@dalwhinnie> show firewall filter vpls-internet-access
Filter: vpls-internet-access
Counters:
Name                                    Bytes           Packets
vpls-internal-attempt                      14868               177

regress@dalwhinnie> show firewall filter vpls-internet-access
Filter: vpls-internet-access
Counters:
Name                                    Bytes           Packets
vpls-internal-attempt                      16464               196

regress@dalwhinnie>
```

And users in oak want to be able to reach the external site at 192.168.120.1/24, shown in Figure A.3.




All core interface IP addresses are in /30 subnets in the 192.168.86/24 network unless otherwise noted.

Figure A.3 External Network Reachability in VPLS via irb Interface

The ISP offering 10.220.90/24 has a BGP session with dalwhinnie. CE4 has reachability:

```
regress@ce4> ping 10.220.90.1
PING 10.220.90.1 (10.220.90.1): 56 data bytes
64 bytes from 10.220.90.1: icmp_seq=0 ttl=63 time=1.002 ms
64 bytes from 10.220.90.1: icmp_seq=1 ttl=63 time=0.984 ms
64 bytes from 10.220.90.1: icmp_seq=2 ttl=63 time=0.984 ms
^C
--- 10.220.90.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.984/0.990/1.002/0.008 ms
regress@ce4>
```

The other sites in oak also have connectivity to 192.168.120.1/24 via the irb.100 interface configured in dalwhinnie's oak instance. CE1 and CE3 can both reach 192.168.120.1/24:

```
regress@ce3> ping 10.220.90.1
PING 10.220.90.1 (10.220.90.1): 56 data bytes
64 bytes from 10.220.90.1: icmp_seq=0 ttl=63 time=1.228 ms
64 bytes from 10.220.90.1: icmp_seq=1 ttl=63 time=30.964 ms
64 bytes from 10.220.90.1: icmp_seq=2 ttl=63 time=1.185 ms
64 bytes from 10.220.90.1: icmp_seq=3 ttl=63 time=1.245 ms
^C
--- 10.220.90.1 ping statistics ---
```

```
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.185/8.655/30.964/12.880 ms

regress@ce1> ping 10.220.90.1
PING 10.220.90.1 (10.220.90.1): 56 data bytes
64 bytes from 10.220.90.1: icmp_seq=0 ttl=63 time=1.207 ms
64 bytes from 10.220.90.1: icmp_seq=1 ttl=63 time=1.172 ms
64 bytes from 10.220.90.1: icmp_seq=2 ttl=63 time=1.171 ms
64 bytes from 10.220.90.1: icmp_seq=3 ttl=63 time=1.178 ms
^C
--- 10.220.90.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.171/1.182/1.207/0.015 ms
```

Note that an *irb* interface within a VPLS instance also provides some additional troubleshooting ability because it offers the core network PE router direct access into the specific VPLS domain, and so too, the ability to ping and traceroute within that domain. So along with the VPLS ping, core network routers can now verify both MAC and IP address reachability. Here dalwhinnie verifies connectivity to MAC address 00:05:85:f4:f0:5e at both Layer 2 and Layer 3 in the oak VPN:

```
regress@dalwhinnie> show route forwarding-table vpn oak
 . . .
 . . .
 . . .
Routing table: oak.vpls
VPLS:
Destination       Type RtRef Next hop      Type Index NhRef Netif
default          perm    0             dscd   608    1
00:05:85:f4:f0:5d/48 user    0           indr 1048574    6
                         192.168.86.2  Push 262244, Push 100016(top)   573  2 ge-1/0/2.0
00:05:85:f4:f0:5e/48 user    0           indr 1048574    6
                         192.168.86.2  Push 262244, Push 100016(top)   573  2 ge-1/0/2.0
lsi.1049104       intf    0             indr 1048574    6
                         192.168.86.2  Push 262244, Push 100016(top)   573  2 ge-1/0/2.0
00:17:cb:05:44:5d/48 user    0           ucst   630    5 ge-1/0/0.550
0x30003/51        user    0             comp   631    2
ge-1/0/0.550      intf    0             ucst   630    5 ge-1/0/0.550
0x30001/51        user    0             comp   623    2
0x30002/51        user    0             comp   618    2

regress@dalwhinnie> ping vpls instance oak destination-mac 00:05:85:f4:f0:5e source-ip
192.168.86.1
! -> oban:oak:ge-1/0/6.700
! -> oban:oak:ge-1/0/6.700
! -> oban:oak:ge-1/0/6.700
! -> oban:oak:ge-1/0/6.700
! -> oban:oak:ge-1/0/6.700
--- vpls ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss

regress@dalwhinnie> ping 192.168.90.10
PING 192.168.90.10 (192.168.90.10): 56 data bytes
64 bytes from 192.168.90.10: icmp_seq=0 ttl=64 time=0.890 ms
64 bytes from 192.168.90.10: icmp_seq=1 ttl=64 time=0.843 ms
64 bytes from 192.168.90.10: icmp_seq=2 ttl=64 time=0.859 ms
64 bytes from 192.168.90.10: icmp_seq=3 ttl=64 time=0.815 ms
64 bytes from 192.168.90.10: icmp_seq=4 ttl=64 time=0.848 ms
^C
regress@dalwhinnie> show arp | match 192.168.90.10
00:05:85:f4:f0:5e 192.168.90.10   192.168.90.10          lsi.1049104       none
```

Even if you don't usually have an *irb* interface configured in your VPLS instance, one can be added, temporarily if necessary, just for this additional troubleshooting capability, but it's imperative that it be removed at the end of troubleshooting session because of the additional access it provides to the Internet and to some core internal addresses.

## Summary

VPLS on the MX series routers allows for easier configuration and the ability to easily provide VPLS users access to external routes. In addition, having the option to use an irb interface for troubleshooting purposes (including Layer 3 ping and traceroute) gives network operators additional capability to help quickly diagnose and resolve customer reported problems.

For your reference, here are the configurations for the relevant portions of the dalwhinnie and oban PE routers.

### Dalwhinnie

```
regress@dalwhinnie> show configuration interfaces
ge-1/0/0 {
    vlan-tagging;
    speed 100m;
    encapsulation vlan-vpls;
    unit 550 {
        encapsulation vlan-vpls;
        vlan-id 550;
        family vpls;
    }
}
ge-1/0/2 {
    speed 100m;
    unit 0 {
        family inet {
            address 192.168.86.1/30;
        }
        family mpls;
    }
}
ge-1/0/4 {
    speed 100m;
    unit 0 {
        family inet {
            address 192.168.86.5/30;
        }
        family mpls;
    }
}
ge-1/0/6 {
    speed 100m;
    unit 0 {
        family inet {
            address 192.168.120.1/30;
        }
    }
}
irb {
    unit 100 {
        family inet {
            filter {
```

```
            input vpls-internet-access;
         }
         address 192.168.90.100/24;
      }
   }
}
lo0 {
   unit 0 {
      family inet {
         address 127.0.0.1/32;
         address 10.200.86.1/32;
      }
   }
}
regress@dalwhinnie> show configuration protocols
rsvp {
   interface ge-1/0/2.0;
   interface ge-1/0/4.0;
}
mpls {
   interface ge-1/0/4.0;
   interface ge-1/0/2.0;
}
bgp {
   family inet {
      any;
   }
   family l2vpn {
      signaling;
   }
   group internal {
      type internal;
      local-address 10.200.86.1;
      export next-hop-self;
      neighbor 10.200.86.3;
      neighbor 10.200.86.4;
   }
   group external {
      export [ export-internal oak-vpls-routes ];
      peer-as 65432;
      neighbor 192.168.120.2;
   }
}
ospf {
   traffic-engineering;
   area 0.0.0.0 {
      interface lo0.0 {
         passive;
      }
      interface ge-1/0/4.0;
      interface ge-1/0/2.0;
   }
}
ldp {
   interface ge-1/0/2.0;
   interface ge-1/0/4.0;
}
regress@dalwhinnie> show configuration routing-instances
oak {
   instance-type vpls;
   vlan-id none;
   interface ge-1/0/0.550;
   routing-interface irb.100;
```

```
    route-distinguisher 10.200.86.1:100;
    vrf-target target:300:200;
    protocols {
        vpls {
            site-range 5;
            no-tunnel-services;
            site ce4 {
                site-identifier 4;
                interface ge-1/0/0.550;
            }
        }
    }
}
regress@dalwhinnie> show configuration policy-options
prefix-list internal-routes {
    192.168.86.0/24;
}
policy-statement export-internal {
    from {
        prefix-list internal-routes;
    }
    then accept;
}
policy-statement next-hop-self {
    from protocol bgp;
    then {
        next-hop self;
        accept;
    }
}
policy-statement oak-vpls-routes {
    term 10 {
        from {
            route-filter 192.168.90.0/24 orlonger;
        }
        then accept;
    }
}
regress@dalwhinnie> show configuration firewall
family inet {
    filter vpls-internet-access {
        term 10 {
            from {
                destination-prefix-list {
                    internal-routes;
                }
            }
            then {
                count vpls-internal-attempt;
                discard;
            }
        }
        term 20 {
            then accept;
        }
    }
}
regress@dalwhinnie> show version
Hostname: dalwhinnie
Model: mx80-48t
JUNOS Base OS boot [10.3R1.9]
```

### Oban

```
regress@oban> show configuration interfaces
ge-1/0/0 {
    speed 100m;
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
ge-1/0/2 {
    speed 100m;
    unit 0 {
        family inet {
            address 192.168.86.14/30;
        }
        family mpls;
    }
}
ge-1/0/4 {
    speed 100m;
    unit 0 {
        family inet {
            address 192.168.86.10/30;
        }
        family mpls;
    }
}
ge-1/0/6 {
    vlan-tagging;
    speed 100m;
    encapsulation vlan-vpls;
    unit 700 {
        encapsulation vlan-vpls;
        vlan-id 700;
        family vpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 127.0.0.1/32;
            address 10.200.86.3/32;
        }
    }
}
regress@oban> show configuration protocols
rsvp {
    interface ge-1/0/4.0;
    interface ge-1/0/2.0;
}
mpls {
    interface ge-1/0/4.0;
    interface ge-1/0/2.0;
}
bgp {
    family inet {
        any;
    }
    family l2vpn {
        signaling;
    }
    group internal {
```

```
        type internal;
        local-address 10.200.86.3;
        neighbor 10.200.86.1;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ge-1/0/4.0;
        interface ge-1/0/2.0;
    }
}
ldp {
    interface ge-1/0/2.0;
    interface ge-1/0/4.0;
}
regress@oban> show configuration routing-instances
oak {
    instance-type vpls;
    vlan-id none;
    interface ge-1/0/0.0;
    interface ge-1/0/6.700;
    route-distinguisher 10.200.86.3:100;
    vrf-target target:300:200;
    protocols {
        vpls {
            site-range 5;
            no-tunnel-services;
            site ce1 {
                site-identifier 1;
                interface ge-1/0/0.0;
            }
            site ce3 {
                site-identifier 3;
                interface ge-1/0/6.700;
            }
        }
    }
}
regress@oban# run show version
Hostname: oban
Model: mx80-48t
JUNOS Base OS boot [10.3R1.9]
```

## Appendix B: Junos Automation

Let's step back for a moment from network planning and examine the role that Junos automation can play in implementing and enforcing network standards. Junos automation is an integrated part of the Junos network operating system. It offers a unique ability to, among other things, automate device configuration, enforce configuration standards, create customized commands, periodically check network parameters, and automate response to specific events. For more information on, and references to, the capabilities Junos automation offers, see the last page of this book.

### A Brief Introduction to Junos Automation

Junos automation consists of three types of scripts: *commit*, *op*, and *event*.

Commit scripts run each time a configuration is committed. Commit scripts can be used to enforce configuration standards (such as ensuring each logical and physical interface has a description), prevent catastrophic configuration errors (such as deleting the entire OSPF or BGP stanza), or to reduce the number of statements in complex configurations.

Op scripts are used in operational mode. They can be used to create custom commands (such as customized *show* commands) and to change device configurations (allowing users to change a configuration in a controlled, structured way).

Event scripts are triggered by event-options policies in response to an event such as a syslog message, SNMP trap, or a custom-timed periodic event. For example, if an OSPF interface goes down, an event script can respond in a consistent, controlled manner to collect specific data, modify the device's configuration, add a custom log message, or all of the above. In short, event-scripts can automate troubleshooting and response to a specific event. Event scripts are closely related to op scripts. In many cases, op scripts can be configured as event scripts and vice versa, the difference being how they are triggered – from the CLI (op script) or from an event (event script).

The uses and capabilities of each type of script and their ability to act together goes well beyond this short introduction, of course, so reference the Day One library and the resources cited on the last page of this book. There is even a Day One automation book that covers the use of XML to effectively and efficiently leverage Junos' native XML capabilities for automation.

But insteading of just telling you about the glory of Junos automation, let's show you.

### An Automation Example

Let's create an example of Junos automation to use for an OSPF link's latency and to set its OSPF metric, say if the OSPF interface has a /30 or /31 network mask and has a specific configuration tag.

The plan would be to send a series of 15 RPM probes out over a defined testing period, ranging from 60 to 2592000 seconds, inclusive. Each time per testing period, the event-script checks the results of the latest completed set of probe results, and reads the average probe latency. If an interface probe's average latency is greater than the allowed variance, then the script modifies that OSPF interface's metric. The example would need to consist of two separate scripts – a commit script and an event script.

## The Commit Script

In general, the commit script keeps the device configuration in line. The commit script performs initial configuration of the solution (the user only need configure the commit and event scripts) and then ensures that there is not an accidental configuration change that breaks it. The commit script does the following upon each commit:

1. Determines if the test-interval defined in the script lies within the valid range 60…2592000 seconds inclusive.
2. For each interface configured in OSPF with apply-macro of *monitor-link-latency*, the commit script checks to see if that interface has a /30 or /31 network mask. If so, it:

   a. Determines the IP address for the remote side of the /30 or /31 OSPF link.

   b. Configures an RPM probe with a target address of the link's remote side (probe name is ospf-metric-probe-<logical-interface>).
3. If an interface is removed from OSPF, and/or has the apply-macro of monitor-link-latency removed, the commit script deletes that interface's solution RPM probes from the configuration.
4. Configures a generated event (under the event-options hierarchy) that occurs at the defined test-interval once per testing period.
5. Configures an event-options policy that triggers upon the generated event and runs this solution's event script.
6. Notifies the user at the CLI and in the logs of any changes it makes to the configuration.
7. If the commit script's $test-interval variable has been modified, the commit script creates a new generated event with that interval, modifies the event-options policy to trigger upon that event, and modifies the RPM probes for each OSPF interface to send an RPM probe at an interval closest to (test-interval)/15 seconds.

The commit script has three variables that can be modified by the user. The most important one is the *$test-interval* variable. Modifying this value allows the user to change the length of the test interval, measured in seconds. Since 15 RPM probes for each OSPF interface probe are sent in a given test interval, modifying this value causes the commit script to modify the probe-interval for each probe whose name contains the regex *ospf-metric-probe-*. Changing this value also causes a new *event-options* generated event with the same value, as well as a modification of the event-options policy *monitor-ospf-interface-probes,* to trigger on the corresponding generated event. In other words, if the user changes this variable, the script will automatically make all the necessary changes to the solution – the user need not make any further changes.

The other two variables that can be modified by the user are *$cmt-script-name* and *$event-script-name,* which is the name of this commit script and the name of the corresponding event script, respectively. The $event-script-name variable is used to ensure that the event-options policy that runs the event script actually is configured for the correct event script. The $cmt-script-name variable is the name of this commit script and is used to identify this script's output in the system's logs.

Any action taken by the commit script is logged and displayed at the CLI upon commit.

## The Event Script

The event-script portion of this solution is activated by an event-options policy configured by the commit script. The event script performs the following tasks:

1. Reads the RPM probe data only for the RPM probes whose names contain the *regex ospf-metric-probe-* and finds the average probe latency for the most recently completed test.
2. Determines a proposed metric for each interface based on its most recent probe latency: metric = (average delay in microsec) /1000 * 10, rounded to nearest integer. In other words, 10*(average delay in msec) rounded to nearest integer.
3. Reads the current metric for each OSPF interface.
4. Determines if the new metric is within a specified variance of the existing metric.

   a. If variance is set to 40%, if the new metric is within 40% of the existing metric, the interface's metric is not changed.

   b. Only if the new metric is more than 40% of the existing metric will that specific interface's metric be changed.
5. Determines the area for each OSPF interface.
6. Changes the metric for each interface if the new metric is outside the allowed variance.
7. Each time the event script runs, it does the following:

   a. Notifies the user via logs if it is making any changes and what those changes are:

   The event script makes configuration changes via an exclusive commit, so if the configuration database is already modified, when the script tries to lock the database, the script's configuration lock will fail and it will log a message describing the reason for the configuration lock fail.

   b. Logs the probe results for each monitored OSPF interface.

   c. If no changes are made to the OSPF metrics, the script logs a message to that effect.

And the event script has three variables that can be modified by the user: *$variance*, *$event-script-name*, and *$metric-up-on-loss*.

1. $variance is a decimal value for the percentage of variance that an interface's RPM probes can tolerate before the event script will modify its OSPF metric. The script is set to provide a metric of 10 for each ms of round trip latency. For example, when $variance = 0.4 (40%), if an interface's metric is 500 and the most recent probe results show an average of 65,000 microseconds (us), the script will not modify the OSPF metric (65,000us = 65 ms = 650 metric; 30% variance). If a new set of results comes in showing a 75,000 us average (50% difference), then the script modifies the OSPF metric to 750.
2. $script-name is the name of this event-script and is used to identify this script's messages in the system logs.
3. $metric-up-on-loss, when set to a value of 1, will metric up the appropriate OSPF interface to 50,000 if any of its RPM probes are lost on the most recent time-interval. If $metric-up-on-loss is set to any value other than 1, then the

interface metric is determined from the available RPM probe data for that interface.

NOTE Any action taken by the event script is logged.

Junos automation offers some powerful benefits, but before deploying any automated solution, it is very important to understand how it works and if it is appropriate for your specific network environment – the last thing you want to do is automate mistakes! With that being said, if this scripting solution is something that may be of use, it's highly recommended that you test it in a lab, if only to become familiar with how it works, so its effect on your production network can be understood and predictable. The settings for *variance* and *test-interval* should be carefully considered, striking the right balance between traffic optimization and keeping churn (unnecessary commits and metric changes) to a minimum. These settings need to be considered for your specific network. For example, setting the variance too low can result in a higher number of commits and OSPF metric changes that don't necessarily alter the path of the traffic – ask yourself *does changing an interface's metric from 500 to 600 necessarily affect traffic patterns*?

## Applying the Automation Solution

The first step is to deploy the scripts to the Junos device. Commit scripts must be placed in the /var/db/scripts/commit/ directory; event scripts must be in the /var/db/scripts/event/ directory. Here is an example of applying the automation to a router for the first time. First let's see what already exists:

```
ps@lagavulin> show configuration protocols ospf
traffic-engineering {
   shortcuts;
}
area 0.0.0.0 {
   interface ge-0/0/2.0 {
      interface-type p2p;
   }
   interface ge-0/0/3.0 {
      interface-type p2p;
   }
   interface lo0.0;
}

ps@lagavulin> show interfaces terse | match ge-0/0 | match inet
ge-0/0/0.0            up    up   inet     172.19.112.200/27
ge-0/0/1.201          up    up   inet     192.168.66.1/30
ge-0/0/1.400          up    up   inet     192.168.90.17/30
ge-0/0/2.0            up    up   inet     192.168.86.2/30
ge-0/0/3.0            up    up   inet     192.168.86.29/30

ps@lagavulin> show configuration services rpm

ps@lagavulin> show configuration event-options

ps@lagavulin>
```

The output shows that no RPM probes or event-options configurations exist. There are two configured OSPF interfaces with /30 masks but they do not have the required apply-macro. So now, let's configure the solution:

```
ps@lagavulin> edit
Entering configuration mode
```

```
[edit]
ps@lagavulin# set system scripts commit file create-ospf-probes-and-evt-
optns-cmt-script.slax

[edit]
ps@lagavulin# set event-options event-script file read-ospf-probes-evt-
script.slax

[edit]
ps@lagavulin# edit protocols ospf

[edit protocols ospf]
ps@lagavulin# show
traffic-engineering {
   shortcuts;
}
area 0.0.0.0 {
   interface ge-0/0/2.0 {
      interface-type p2p;
   }
   interface ge-0/0/3.0 {
      interface-type p2p;
   }
   interface lo0.0;
}

[edit protocols ospf]
ps@lagavulin# set area 0 interface ge-0/0/2.0 apply-macro monitor-link-
latency

[edit protocols ospf]
ps@lagavulin# set area 0 interface ge-0/0/3.0 apply-macro monitor-link-
latency

[edit protocols ospf]
ps@lagavulin# show
traffic-engineering {
   shortcuts;
}
area 0.0.0.0 {
   interface ge-0/0/2.0 {
      apply-macro monitor-link-latency;
      interface-type p2p;
   }
   interface ge-0/0/3.0 {
      apply-macro monitor-link-latency;
      interface-type p2p;
   }
   interface lo0.0;
}

[edit protocols ospf]
ps@lagavulin# commit
warning: commit script create-ospf-probes-and-evt-optns-cmt-script.slax
adding event-options generated-event 600-seconds to configuration to trigger
OSPF RPM probe monitoring
warning: commit script create-ospf-probes-and-evt-optns-cmt-script.slax
adding event-options policy 'monitor-ospf-interface-probes' to configuration
to trigger OSPF RPM probe monitoring event-script.
warning: commit script create-ospf-probes-and-evt-optns-cmt-script.slax
adding rpm probe ospf-metric-probe-ge-0/0/2.0 in response to interface
ge-0/0/2.0 being in ospf, having a /30 or /31 netmask, and having an apply-
```

```
macro of 'monitor-link-latency'
warning: commit script create-ospf-probes-and-evt-optns-cmt-script.slax
adding rpm probe ospf-metric-probe-ge-0/0/3.0 in response to interface
ge-0/0/3.0 being in ospf, having a /30 or /31 netmask, and having an apply-
macro of 'monitor-link-latency'
commit complete

[edit protocols ospf]
ps@lagavulin# top show services rpm
probe ospf-metric-probe-ge-0/0/2.0 {
    test 192.168.86.1 {
        probe-type icmp-ping;
        target address 192.168.86.1;
        probe-count 15;
        probe-interval 40;
        test-interval 10;
    }
}
probe ospf-metric-probe-ge-0/0/3.0 {
    test 192.168.86.30 {
        probe-type icmp-ping;
        target address 192.168.86.30;
        probe-count 15;
        probe-interval 40;
        test-interval 10;
    }
}

[edit protocols ospf]
ps@lagavulin# top show event-options
generate-event {
    600-seconds time-interval 600;
}
policy monitor-ospf-interface-probes {
    events 600-seconds;
    then {
        event-script read-ospf-probes-evt-script.slax;
    }
}
event-script {
    file read-ospf-probes-evt-script.slax;
}
```

Once the event script and commit script are configured and the desired OSPF interfaces are tagged with the needed apply-macros (in this case, both of the interfaces get tagged for monitoring), upon commit the commit script configures the necessary RPM probes, event-options policy, and generated events. The commit script notifies the user of the configuration changes with a CLI message and an equivalent message in the logs.

It's up and operational!

NOTE The *apply-macro* configuration is a hidden configuration, so it must be typed out (it cannot be tab-completed).

NOTE When an interface is initially added to monitoring, the first few log messages with the probe results for that interface may show potential packet loss due to loss of NaN probes. It's expected behavior since the newly added interface may not have a full set of RPM data from which to determine an average latency. Here's an example:

```
Feb 10 04:59:32  dalwhinnie cscript: message from event-script read-ospf-
probes-evt-script.slax - OSPF RPM PROBE ospf-metric-probe-ge-0/0/3.0 lost
NaN probe(s) during its most recent run.  Verify that no packet loss is
occurring on ge-0/0/3.0
```

And now, here is an example of an interface being added to OSPF link latency monitoring – our automation solution is already operational on the router on ge-0/0/2.0. You can see that the ge-0/0/3.0 interface is already configured in OSPF but is not monitored:

```
[edit protocols ospf]
ps@dalwhinnie# top show services rpm
probe ospf-metric-probe-ge-0/0/2.0 {
   test 192.168.86.5 {
      probe-type icmp-ping;
      target address 192.168.86.5;
      probe-count 15;
      probe-interval 40;
      test-interval 10;
   }
}

[edit protocols ospf]
ps@dalwhinnie# show
traffic-engineering {
   shortcuts;
}
area 0.0.0.0 {
   interface ge-0/0/2.0 {
      apply-macro monitor-link-latency;
      interface-type p2p;
      metric 21;
   }
   interface ge-0/0/3.0 {
      metric 19;
   }
   interface lo0.0;
}

[edit protocols ospf]
ps@dalwhinnie# set area 0 interface ge-0/0/3.0 apply-macro monitor-link-
latency

[edit protocols ospf]
ps@dalwhinnie# commit
warning: commit script create-ospf-probes-and-evt-optns-cmt-script.slax
adding rpm probe ospf-metric-probe-ge-0/0/3.0 in response to interface
ge-0/0/3.0 being in ospf, having a /30 or /31 netmask, and having an apply-
macro of 'monitor-link-latency'
commit complete

[edit protocols ospf]
ps@dalwhinnie# show
traffic-engineering {
   shortcuts;
}
area 0.0.0.0 {
```

```
    interface ge-0/0/2.0 {
        apply-macro monitor-link-latency;
        interface-type p2p;
        metric 21;
    }
    interface ge-0/0/3.0 {
        apply-macro monitor-link-latency;
        metric 19;
    }
    interface lo0.0;
}

[edit protocols ospf]
ps@dalwhinnie# top show services rpm
probe ospf-metric-probe-ge-0/0/2.0 {
    test 192.168.86.5 {
        probe-type icmp-ping;
        target address 192.168.86.5;
        probe-count 15;
        probe-interval 40;
        test-interval 10;
    }
}
probe ospf-metric-probe-ge-0/0/3.0 {
    test 192.168.86.29 {
        probe-type icmp-ping;
        target address 192.168.86.29;
        probe-count 15;
        probe-interval 40;
        test-interval 10;
    }
}
```

In this output example, ge-0/0/3.0 has a /30 network mask (not shown) and is configured in OSPF but not monitored by this solution. Addition of the appropriate *apply-macro* configuration causes the commit script to add an RPM probe for that link: it is now monitored by our automation solution and its metric will be modified, if necessary, by it. The commit script notifies the user of the config changes being made with a CLI message (shown) and with the same log message (not shown).

The event script checks the last complete set of probe results one time for every desired test-interval and logs the probe results for each interface and any configuration changes, or lack of changes, by our automation example:

```
Feb 10 05:48:01  lagavulin cscript: $avg-delay for interface ge-0/0/2.0 =  2173 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 05:48:01  lagavulin cscript: event-script read-ospf-probes-evt-script.slax changing ospf
interface ge-0/0/2.0 metric from  to 22 based on rpm probe data.
Feb 10 05:48:01  lagavulin cscript: $avg-delay for interface ge-0/0/3.0 =  2297 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 05:48:01  lagavulin cscript: event-script read-ospf-probes-evt-script.slax changing ospf
interface ge-0/0/3.0 metric from  to 23 based on rpm probe data.
Feb 10 05:48:10  lagavulin cscript: configuration change to modify ospf metric values successful.
this msg generated by read-ospf-probes-evt-script.slax
Feb 10 05:58:01  lagavulin cscript: $avg-delay for interface ge-0/0/2.0 =  2300 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 05:58:01  lagavulin cscript: $avg-delay for interface ge-0/0/3.0 =  1938 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 05:58:01  lagavulin cscript: this message generated by read-ospf-probes-evt-script.slax:
ospf probe results show no change in link latency outside the allowed variance of 40%.
```

Here, during the interval ending 05:48:01, the event script logged the RPM probe results for each interface and changed the metric for each monitored interface from default to the value reflected by the RPM probe latency for that interface. At the end of the interval ending 05:58:01, the event script logged the probe results and reported no probe results were outside the required 40% variance of the current interface metrics.

And here, for your perusal, is a log snippet from the event-script for a metric modification for interface ge-0/0/3.0:

```
Feb 10 06:28:01  lagavulin cscript: $avg-delay for interface ge-0/0/2.0 =  2642 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 06:28:01  lagavulin cscript: $avg-delay for interface ge-0/0/3.0 =  3357 usec; this msg
generaged by read-ospf-probes-evt-script.slax
Feb 10 06:28:01  lagavulin cscript: event-script read-ospf-probes-evt-script.slax changing ospf
interface ge-0/0/3.0 metric from 23 to 34 based on rpm probe data.
Feb 10 06:28:10  lagavulin cscript: configuration change to modify ospf metric values successful.
this msg generated by read-ospf-probes-evt-script.slax
```

## Potential Uses of the Automation Example

There are several potential uses of our example, three of which are:

1. Layer 1 and Layer 2 services from a SP. If your company uses Layer 1 transport or Layer 2 services such as VPLS or point-to-point transport from a service provider (SP), you have no control over the actual path the traffic takes once it reaches the SP's network. A fiber incident, maintenance, or outage for any other reason on the SP network can affect the path the traffic takes, greatly changing the latency. Our automation example can be used on OSPF interfaces for which you don't control the physical path, or for situations otherwise where the link latency is likely to have wide variations. Simply ensuring the necessary interfaces have a /30 or /31 netmask, tagging them with the apply-macro value monitor-link-latency, and then configuring the event and commit scripts will get the automation solution up and running on those interfaces.
2. Monitoring only. If your network latency is steady state and the latencies will not change radically (you have direct control over all your point-to-point links), you can use this automation example to set up appropriate latency-based OSPF metrics for all OSPF interfaces, and then log latency and packet loss results. Set the variance in the event script to extremely high (10 = 1000%) and the test-interval in the commit script to very short (60 sec). This implementation sets link metrics according to latency and then logs link latency and potential packet loss every 60 seconds. The high variance ensures that link metrics won't change, keeping traffic patterns consistent. It's a useful implementation if only monitoring for potential packet loss and actual link latency is important.
3. A building block. This automation example shows a moderately complex scripting solution. A more experienced Junos Automation user or someone learning can use this script in a lab as a starting point, making modifications for production or simply learning the ropes.

## Scripts

The commit script and the event script that comprise our automation example are reprinted here in their entirety. The first is the commit script, and the event script follows.

TIP If you visit www.juniper.net/dayone, and then follow the path to this book's download page, you'll find a free *Copy and Paste* edition of this book. Its rich-text format allows the file to be opened in various text editors for easy copying and pasting of the book's configurations.

## create-ospf-probes-and-evt-optns-cmt-script.slax

```
version 1.0;

ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";

import "../import/junos.xsl";



/* YOU MUST ACCEPT THE TERMS OF THIS DISCLAIMER TO USE THIS SOFTWARE.
*
* JUNIPER IS WILLING TO MAKE THE INCLUDED SCRIPTING SOFTWARE AVAILABLE TO YOU ONLY UPON THE
CONDITION THAT YOU  * ACCEPT ALL OF THE TERMS CONTAINED IN THIS DISCLAIMER. PLEASE READ THE TERMS
AND CONDITIONS OF THIS DISCLAIMER * CAREFULLY.

* THE SOFTWARE CONTAINED IN THIS FILE IS PROVIDED "AS IS." JUNIPER MAKES NO WARRANTIES OF ANY KIND
WHATSOEVER
* WITH RESPECT TO SOFTWARE. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
* WARRANTY OF NON-INFRINGEMENT OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE,
ARE HEREBY
* DISCLAIMED AND EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW.
*
* IN NO EVENT WILL JUNIPER BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, SPECIAL,
INDIRECT,
* CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF
LIABILITY ARISING
* OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF JUNIPER HAS BEEN ADVISED OF THE
POSSIBILITY OF
* SUCH DAMAGES. */

/* an explanation of this script's functionality lies at the bottom of this file */



/* created by Tim Fiola, Juniper Networks Professional Services*/

/*****************************************************************/
/* these first 3 variables are all that need to be modified if  */
/* the user wishes to modify the script parameters;       */
/* any other mods require a good working knowledge of junos */
/* automation                          */
/*****************************************************************/

/* cmt-script-name is the name of this commit script  */
/* this is used in the log messages to ID the    */
/* specific script that is creating the message   */
var $cmt-script-name = "create-ospf-probes-and-evt-optns-cmt-script.slax";

/* event-script-name is the name of the paired event-script that */
/* gets triggered every test-interval and reads the rpm probe data */
```

```
/* this needs to be accurate to ensure that the event-options     */
/* policy gets created properly    */
var $event-script-name = "read-ospf-probes-evt-script.slax";

/* interval is how long each monitoring period is; 15 probes  */
/* are sent every monitoring period; enter time in seconds    */
/* this value must be be in range 60..2592000 seconds inclusive */
var $test-interval = 60;

/* MODIFYING ANYTHING OTHER THAN THE THREE VARIABLES ABOVE REQUIRES A  */
/* GOOD WORKING KNOWLEDGE OF JUNOS AUTOMATION AND IS NOT RECOMMENDED */


match configuration {

          /* task 0 - determine if $test-interval is valid*/
          if ($test-interval < 60 or $test-interval > 2592000) {
            var $test-interval-message = "the commit script " _ $cmt-script-name _ " test interval
must be be in range 60..2592000 seconds inclusive; the configured test-interval is " _ $test-
interval;
             <xnm:error> {
               <xsl:message terminate="yes"> $test-interval-message;
             }
          }

          /* task 1 - determine OSPF interfaces */
          var $ospf-interfaces := {
            call ospf-interfaces($configuration = .);
          }

          /* debug output*/
           <xnm:warning> {
            <message> {
              /* expr "ospf-interfaces is "; */
              copy-of $ospf-interfaces;
            }
          }

          /* task 2 - determine the destination IP for each interface */
          var $dest-ip := {
            call dest-ip($ospf-interfaces, $configuration = .);
          }

          /* debug output*/
           <xnm:warning> {
            <message> {
              /* expr "dest-ip is "; */
              copy-of $dest-ip;
            }
          }

          /* task 3 - configure the probe for each interface and the*/
          /* event-options generated-event and policy         */
          var $num-probes = count($dest-ip/dest-ip);

           /* <xnm:warning> {
          * <message> {
          *   expr "num-probes = " _ $num-probes;
          * }
          *} */
          call create-probes-events($num-probes, $dest-ip, $configuration = ., $ospf-interfaces);
```

```
}

/*****************************************************************************/
/* this template determines which interfaces are configured in OSPF and builds   */
/* an element containing all of the interface names configured in OSPF      */
/*****************************************************************************/
template ospf-interfaces($configuration) {
         for-each ($configuration/protocols/ospf/area/interface[apply-macro/name == "monitor-
link-latency"]) {
           var $physical-int = substring-before(name, "\.");
           var $logical-int = substring-after(name, "\.");

           /* if the int is configured in OSPF with the apply-macro    */
           /* and has a /30 or /31 mask, add the interface to the element*/
           if ($configuration/interfaces/interface[name == $physical-int]/unit[name == $logical-
int]/family/inet/address[contains(name, "\/30") or contains(name, "\/31")]) {
             copy-of name;
           }
         }
}

/*****************************************************************************************/
/* this template determines the destination ip address for the needed RPM probes and places  */
/* those values within an element that stores each value as text in a <dest-ip> node     */
/*****************************************************************************************/
template dest-ip($ospf-interfaces, $configuration) {
         for-each ($ospf-interfaces/name) {
           var $physical-int = substring-before(., "\.");
           var $logical-int = substring-after(., "\.");
           for-each ($configuration/interfaces/interface[name == $physical-int]/unit[name ==
$logical-int]/family/inet/address[contains(name, "\/30") or contains(name, "\/31")]/name) {
             var $parsed-address := jcs:parse-ip(.);
             var $host-ip = $parsed-address[1]; /* takes the first element of the parse IP, which
is the address  */

             /* separate the address into its component octets */
             var $first-octet = substring-before($host-ip, "\.");
             var $second-octet = substring-before(substring-after($host-ip, "\."), "\.");
             var $third-octet = substring-before(substring-after(substring-after($host-ip, "\."),
"\."), "\.");
             var $last-octet = substring-after(substring-after(substring-after($host-ip, "\."),
"\."), "\.");

             /* determine the destination IP address, derived from the host address */
             var $last-octet-dest-ip = {
               if($parsed-address[3] = 30) {
                 if(($last-octet)mod(4) == 1) {
                   expr $last-octet + 1;
                 } else if (($last-octet)mod(4) == 2) {
                   expr $last-octet - 1;
                 } else {
                   expr "9999";  /* dummy value showing that address is ineligible */
                 }
               } else if ($parsed-address[3] = 31){
                 if(($last-octet)mod(2) == 0) {
                   expr $last-octet + 1;
                 } else if (($last-octet)mod(2) == 1) {
                   expr $last-octet - 1;
                 } else {
                   expr "9999";   /* dummy value showing that address is ineligible */
                 }
               }
```

```
            }

            /* create the full destination IP address using the last octet of   */
            /* the destination address and add it to the dest-ip-element element*/
            var $dest-ip = $first-octet _ "." _ $second-octet _ "." _ $third-octet _ "." _ $last-
octet-dest-ip;
            var $dest-ip-element := {
              <dest-ip> $dest-ip;
            }
            copy-of $dest-ip-element;
          }
        }
}

/*****************************************************************************************/
/* this template creates the change elements that configure the necessary event-options */
/* policy and generated events if they are not already present; it also creates the RPM */
/* probes for the ospf interfaces if they are not present.  it will delete any           */
/* configured rpm probes for ospf interfaces that have since been removed from ospf      */
/*****************************************************************************************/
template create-probes-events($num-probes, $dest-ip, $configuration, $ospf-interfaces) {

        /* event-name is the name of the generated event that will trigger the event-script    */
        var $event-name = $test-interval _ "-seconds";

        /* this is the actual change element */
        var $change := {

          /* this part looks for the necessary generated event  */
          if(not($configuration/event-options/generate-event[time-interval == $test-interval &&
name == $event-name])) {
            <event-options> {
              <generate-event> {
                <name> $event-name;
                <time-interval> $test-interval;
              }
            }
          }

          /* this part looks for the necessary event-option policy, triggered by   */
          /* the generated event; this policy executes the event-script that       */
          /* reads the rpm probe data  */
          <event-options> {
            if (event-options/policy[name == "monitor-ospf-interface-probes"]) {
              <policy delete="delete"> { /* delete the policy before adding it back with any
modified info*/
                <name> "monitor-ospf-interface-probes";
              }
            }
            <policy> {
              <name> "monitor-ospf-interface-probes";
              <events> $event-name;
              <then> {
                <event-script> {
                  <name> $event-script-name;
                }
              }
            }
          }
```

```
            /* this section adds necessary probes to the config change element if they don't exist
*/
            /* or if the needed rpm probes have been manually modified in a way that would break  */
            /* the solution, this section corrects those manual config errors                     */
            for-each($ospf-interfaces/name) {
              var $probe-name = "ospf-metric-probe-" _ .;

              /* determine the numbered position of current 'name' element within $ospf-interfaces
node      */
              /* this is the position of the current ospf int name within the $ospf-interfaces array
*/
              var $position = position();

              /* the test-name will be the probe's destination IP address */
              var $test-name = $dest-ip/dest-ip[$position];

              /* probe-interval is the test-interval/15 probes (15 probes sent/test-interval)   */
              var $probe-interval = round(($test-interval)div(15));

              /* if there is not a probe configured for the ospf int */
              if(not($configuration/services/rpm/probe[name == $probe-name])) {

                <services> {
                  <rpm> {
                    <probe> {
                      <name> $probe-name;
                      <test> {
                        <name> $test-name;
                        <probe-type> "icmp-ping";
                        <target> {
                          <address> $test-name;
                        }
                        <probe-count> 15;
                        <probe-interval> $probe-interval;
                        <test-interval> 10;
                      }
                    }
                  }
                }
              }

              /* if there is a probe configured for the ospf int but is has incorrect configs */
              if (($configuration/services/rpm/probe[name == $probe-name]/test[name != $test-name
or target/address != $test-name or probe-interval != $probe-interval]) or ($configuration/
services/rpm/probe[name == $probe-name]/test[not(name) or not(probe-type) or not(probe-count) or
not(probe-interval) or not(test-interval)])) {
                <services> {
                  <rpm> {
                    <probe delete="delete"> {
                      <name> $probe-name;
                    }
                    <probe> {
                      <name> $probe-name;
                      <test> {
                        <name> $test-name;
                        <probe-type> "icmp-ping";
                        <target> {
                          <address> $test-name;
                        }
                        <probe-count> 15;
                        <probe-interval> $probe-interval;
                        <test-interval> 10;
```

```
                }
              }
            }
          }
        }

        /* this section deletes probes for interfaces that are not in OSPF anymore   */
        for-each($configuration/services/rpm/probe[contains(name, "ospf-metric-probe-")]) {
          var $int = substring-after(name, "ospf-metric-probe-");
          if(not(contains($ospf-interfaces, $int))) {
            var $probe-name = name;
            <services> {
              <rpm> {
                <probe delete="delete"> {
                  <name> $probe-name;
                }
              }
            }
          }
        }
      }


      /* these next 4 loops use the same for-each & if/then logic as above to send status */
      /* output to CLI and to the logs; these could not be part of the above statements   */
      /* or they'd have created undesirable nodes in the change element */

      /* 1 - if generated event has been added by this script  */
      if(not($configuration/event-options/generate-event[time-interval == $test-interval &&
name == $event-name])) {
        var $status-msg-event = "commit script " _ $cmt-script-name _ " adding event-options
generated-event " _ $event-name _ " to configuration to trigger OSPF RPM probe monitoring";
        call log-status-msg($status-msg = $status-msg-event);
        <xnm:warning> {
          <message> {
            expr $status-msg-event;
          }
        }

      }

      /* 2 - if event-options policy has been added by this script*/
      if(not($configuration/event-options/policy[(events == $event-name) && (name == "monitor-
ospf-interface-probes")]/then/event-script[name == $event-script-name])) {
        var $status-msg-policy = "commit script " _ $cmt-script-name _ " adding event-options
policy 'monitor-ospf-interface-probes' to configuration to trigger OSPF RPM probe monitoring event-
script.";
        call log-status-msg($status-msg = $status-msg-policy);
        <xnm:warning> {
          <message> {
            expr $status-msg-policy;
          }
        }
      }

      /* 3 - if RPM probe(s) have been added by this script or if the needed rpm probes   */
      /* have been manually modified in a way that would break the solution, this        */
      /* section corrects those manual config errors                    */
      for-each($ospf-interfaces/name) {

        var $probe-name = "ospf-metric-probe-" _ .;
```

```
            /* determine the numbered position of current 'name' element within $ospf-interfaces
node      */
            /* this is the position of the current ospf int name within the $ospf-interfaces array
*/
            var $position = position();

            /* the test-name will be the probe's destination IP address */
            var $test-name = $dest-ip/dest-ip[$position];

            /* probe-interval is the test-interval/15 probes (15 probes sent/test-interval) */
            var $probe-interval = ($test-interval)div(15);

            if(not($configuration/services/rpm/probe[name == $probe-name])) {
              var $status-msg = "commit script " _ $cmt-script-name _ " adding rpm probe " _ $probe-
name _ " in response to interface " _ . _ " being in ospf, having a /30 or /31 netmask, and having an
apply-macro of 'monitor-link-latency'";
              call log-status-msg($status-msg);
              <xnm:warning> {
                <message> {
                  expr $status-msg;
                }
              }
            }

            if (($configuration/services/rpm/probe[name == $probe-name]/test[name != $test-name or
target/address != $test-name or probe-interval != $probe-interval]) or ($configuration/services/
rpm/probe[name == $probe-name]/test[not(name) or not(probe-type) or not(probe-count) or not(probe-
interval) or not(test-interval)])) {
              var $status-msg = "commit script " _ $cmt-script-name _ " modifying rpm probe " _
$probe-name _ " in order to maintain required probe configuration";
              call log-status-msg($status-msg);
              <xnm:warning> {
                <message> {
                  expr $status-msg;
                }
              }
            }
          }

          /* 4 - if this script is deleting RPM probes for interfaces that are not in OSPF anymore
*/
          for-each($configuration/services/rpm/probe[contains(name, "ospf-metric-probe-")]) {
            var $int = substring-after(name, "ospf-metric-probe-");
            if(not(contains($ospf-interfaces, $int))) {
              var $probe-name = name;
              var $status-msg = "commit script " _ $cmt-script-name _ " removing rpm probe " _
$probe-name _ " in response to interface " _ $int _ " being removed from ospf, not being designated
as a monitored ospf interface, or having a network mask that is not /30 or /31";
              call log-status-msg($status-msg);
              <xnm:warning> {
                <message> {
                  expr $status-msg;
                }
              }
            }
          }


          /* debug output*/
          <xnm:warning> {
            <message> {
              copy-of $change;
            }
```

```
        }

        /* make the configuration change */
        call jcs:emit-change($dot = ., $content = $change);


}

/*************************************************/
/* this template puts $status-msg in the log */
/*************************************************/
template log-status-msg($status-msg) {
        expr jcs:syslog(5, $status-msg);
}



/* This automation solution sends a series of 15 RPM probes out over a defined testing period,
ranging from 60
* to 2592000 seconds, inclusive.  One time per testing period, the event-script checks the results of
the latest
* completed set of probe results and reads the average probe latency. If an interface probe's average
latency is
* greater than the allowed variance, then the script modifies that OSPF interface's metric. This
solution
* consists of two separate scripts – a commit script and an event script.
*
* In general, the commit script keeps the device configuration in line. The commit script performs
initial
* configuration of this solution (the user only need configure the commit and event scripts) and then
ensures
* that there is not an accidental configuration change that breaks it. The commit script does the
following upon
* each commit:
* 1)      Determines if the test-interval defined in the script lies within the valid range
60..2592000 seconds
* inclusive
* 2)      For each interface configured in OSPF with an apply-macro set to value 'monitor-link-
latency', the
* commit script checks to see if that interface has a /30 or /31 network mask. This solution only
runs
* on ospf interfaces with the apply-macro value 'monitor-link-latency'
* 3)      Determines the IP address for the remote side of the /30 or /31 OSPF link
* 4)      Configures an RPM probe with a target address of the link's remote side (probe name is
* ospf-metric-probe-<logical-interface>)
* 5)      If an interface is removed from OSPF, the commit script deletes that interface's solution
RPM probes
* from the configuration
* 6)      Configures a generated event (under the event-options hierarchy) that occurs at the
defined test
* interval once per testing period
* 7)      Configures an event-options policy that triggers upon the generated event and runs this
solution's event
* script
* 8)      Notifies the user at the CLI and in the logs of any changes it makes to the configuration
* 9)      If the commit script's $test-interval variable has been modified, the commit script
creates a new
* generated event with that interval, modifies the event-options policy to trigger upon that event,
and modifies
* the RPM probes for each OSPF interface to send an RPM probe at an interval close to (test-
interval)/15 seconds
*
```

```
*
* The commit script has three variables that can be modified by the user. The most important one is
the
* $test-interval variable. Modifying this value allows the user to change the length of the test
interval. This
* value is the length of the test interval, in seconds. Since 15 RPM probes for each OSPF interface
probe are
* sent in a given test interval, modifying this value causes the commit script to modify the probe-
interval for
* each probe whose name contains the regex 'ospf-metric-probe-'. Changing this value also causes a
new
* event-options generated event with the same value and a modification of the event-options policy
* 'monitor-ospf-interface-probes' to trigger on the corresponding generated event. In other words,
if the user
* changes this variable, the script will automatically make all the necessary changes to the
solution - the user
* need not make any further changes.
* The other two variables, $cmt-script-name and $event-script-name are the names of this commit
script and the
* name of the corresponding event script, respectively. The $event-script-name variable is used to
ensure that
* the event-options policy that runs the event script actually is configured for the correct event
script. The
* $cmt-script-name variable is the name of this commit script and is used to identify this script's
output in
* the system's logs.
*
* any actions taken by the commit script are logged and also noted at the CLI upon commit
* ------------------------------------------------------------------------------------------
-
*
*The event-script portion of this solution is activated by an event-options policy configured by
the commit
* script. The event script performs the following tasks -
* 1)      Reads the RPM probe data only for the RPM probes whose names contain the regex ospf-
metric-probe- and
* finds the average probe latency for the most recently completed test
* 2)      Determines a proposed metric for each interface based on its most recent probe latency
(metric =
* (average delay in microsec)/1000 * 10, rounded to nearest integer.  In other words, 10*(average
delay in msec)
* rounded to nearest integer)
* 3)      Reads the current metric for each OSPF interface
* 4)      Determines if the new metric is within a specified variance of the existing metric.
* a.      If variance is set to 40% (0.40),  if the new metric is within 40% of the existing metric,
the
* interface's metric is not changed
* b.      Only if the new metric is more than 40% of the existing metric will that specific
interface's metric be
* changed
* 5)      Determines the area for each OSPF interface
* 6)      Changes the metric for each interface if the new metric is outside the allowed variance
* 7)      Each time the event script runs, it notifies the user via logs if it is making any changes
and what
* those changes are. If no changes are made, the script logs a message to that effect
*
*
* The event script has three variables that can be modified by the user, $variance,$event-script-
name, and
* $metric-up-on-loss.
*         a) $variance is a decimal value for the percentage of variance that an interface's RPM
probes can tolerate
* before the event script will modify its OSPF metric. The script is set to provide a metric of 10
```

```
for each ms
* of round trip latency. For example, when $variance = 0.4 (40%),  if an interface's metric is 500
and the most
* recent probe results show an average of 65,000 microseconds (us), the script will not modify the
OSPF metric
*  (65,000us = 65 ms = 650 metric; 30% variance). If a new set of results comes in showing a 75,000 us
average
*  (50% difference), then the script modifies the OSPF metric to 750.
*         b) $script-name is the name of this event-script and is used to identify this script's
messages in the
* system logs.
*         c) $metric-up-on-loss, when set to a value of 1, will metric up the appropriate OSPF
interface to 50,000
* if any of its RPM probes are lost on the most recent time-interval. If $metric-up-on-loss is set to
any
* value other than 1, then the interface metric is determined from the available RPM probe data for
that
* interface.
*
* any actions taken by the event-script are logged /*
```

## read-ospf-probes-evt-script.slax

```
version 1.0;

ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
ns ext = "http://xmlsoft.org/XSLT/namespace";

ns math = "http://exslt.org/math";

import "../import/junos.xsl";


/* YOU MUST ACCEPT THE TERMS OF THIS DISCLAIMER TO USE THIS SOFTWARE.
*
* JUNIPER IS WILLING TO MAKE THE INCLUDED SCRIPTING SOFTWARE AVAILABLE TO YOU ONLY UPON THE CONDITION
THAT YOU  * ACCEPT ALL OF THE TERMS CONTAINED IN THIS DISCLAIMER. PLEASE READ THE TERMS AND
CONDITIONS OF THIS DISCLAIMER * CAREFULLY.

* THE SOFTWARE CONTAINED IN THIS FILE IS PROVIDED "AS IS." JUNIPER MAKES NO WARRANTIES OF ANY KIND
WHATSOEVER
* WITH RESPECT TO SOFTWARE. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
* WARRANTY OF NON-INFRINGEMENT OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE,
ARE HEREBY
* DISCLAIMED AND EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW.
*
* IN NO EVENT WILL JUNIPER BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, SPECIAL,
INDIRECT,
* CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF
LIABILITY ARISING
* OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF JUNIPER HAS BEEN ADVISED OF THE
POSSIBILITY OF
* SUCH DAMAGES. */

/* an explanation of this script's functionality lies at the bottom of this file */

/* created by Tim Fiola, Juniper Networks Professional Services*/
```

```
/*************************************************************/
/* these first 3 variables are all that need to be modified if  */
/* the user wishes to modify the script parameters;       */
/* any other mods require a good working knowledge of junos */
/* automation                          */
/*************************************************************/

/* $event-script-name is the script name that appears in the logs with this script's */
/* messages this name will appear in log messages created by this script so that the */
/* user can determine which script is generating a specific log message*/
var $event-script-name = "read-ospf-probes-evt-script.slax";

/* the variance describes the % threshold that will trigger a modification of the    */
/* ospf interface metric; if the new metric differs from the current metric by more  */
/* than this percentage, the ospf interface will get configured with the new metric  */
var $variance = 0.40; /* if the metric difference is greater than this %, modify the metric  */

/* if this is = 1, then if there is a probe loss on the interface the metric will be  */
/* changed to 50000. if it is anything other than 1, then the script will use the probe   */
/* data to determine the metric. if there is no probe loss, this value is not used       */
var $metric-up-on-loss = 1;

/* MODIFYING ANYTHING OTHER THAN THE THREE VARIABLES ABOVE REQUIRES A   */
/* GOOD WORKING KNOWLEDGE OF JUNOS AUTOMATION AND IS NOT RECOMMENDED */


/* opens a connection with mgd for jcs:execute() fn */
var $connection = jcs:open();

/* this sequence grabs the running configuration  */
var $rpc-configuration = <get-configuration>;
var $configuration = jcs:execute($connection, $rpc-configuration);

match / {
   <op-script-output> {

            if(not($connection)) {
              var $log-msg-connection = $event-script-name _ " event-script unable to open
connection to mgd";
              expr jcs:syslog(5, $log-msg-connection );
            }

            /* this sequence grabs the most recent rpm probe results data   */
            var $rpc-get-probe-results = {<command> "show services rpm probe-results";}
            var $probe-results = jcs:execute($connection, $rpc-get-probe-results);

            /* read the data for each probe with 'ospf-metric-probe-' in the owner name  */
            /* and determine if a change to an interface's metric needs to happen     */
            call config-ospf-probe-data($probe-results);


         }
}

/*************************************************************************************************
*/
/* this template reads the rpm probe history and, based on the results, creates any necessary    */
/* change elements for modifying the OSPF interface metric                    */
/*************************************************************************************************
*/
template config-ospf-probe-data($probe-results) {

         var $change := {
```

```
          <configuration> {
            <protocols> {
              <ospf> {
                for-each ($probe-results/probe-test-results[contains(owner, "ospf-metric-
probe-")]) {

                  /* check for probe/packet loss */
                  var $probes-sent = probe-last-test-results/probe-test-generic-results/probes-
sent;
                  var $probe-responses = probe-last-test-results/probe-test-generic-results/
probe-responses;
                  var $difference = $probes-sent - $probe-responses;
                  var $int = substring-after(owner, "ospf-metric-probe-");
                  if($difference != 0) {
                    var $packet-loss-msg = "message from event-script " _ $event-script-name _ "
- OSPF RPM PROBE " _ owner _ " lost " _ $difference _ " probe(s) during its most recent run.  Verify
that no packet loss is occurring on " _ $int;
                    call log-status-msg($status-msg = $packet-loss-msg);
                  }


                  /* determine if the difference between the new metric and the     */
                  /* current-metric is greater than the allowed variance.  if it    */
                  /* is greater, then change to the new-metric.  if it is not    */
                  /* greater (ie - it is within the allowed variance), then do  */
                  /* not change the metric */

                  var $avg-delay = { /* this is a conditional variable*/
                    if (probe-last-test-results/probe-test-generic-results/probe-test-rtt/probe-
summary-results/avg-delay[contains(@junos:format, "usec")]) { /* if the probe results come back in
microseconds */
                      expr probe-last-test-results/probe-test-generic-results/probe-test-rtt/
probe-summary-results/avg-delay;
                  } else if ((probe-last-test-results/probe-test-generic-results/probe-test-rtt/
probe-summary-results/avg-delay[contains(@junos:format, "msec")])) { /* if the probe results come
back in milliseconds */
                      expr (probe-last-test-results/probe-test-generic-results/probe-test-rtt/
probe-summary-results/avg-delay)*1000;
                    }
                  }

                  var $metric-msg = "$avg-delay for interface " _ $int _ " = " _ $avg-delay _ "usec;
this msg generaged by " _ $event-script-name;
                  call log-status-msg($status-msg = $metric-msg);

                  var $int-area := {
                    call ospf-int-area($int);
                  }
                  var $new-metric = {
                    call determine-metric($avg-delay, $difference);
                  }
                  var $current-metric = $configuration/protocols/ospf/area/interface[name = $int]/
metric;
                  if(((math:abs($new-metric - $current-metric)div($current-metric)) > $variance)
or not($current-metric)) {
                    /* if there is no average delay info for the last-probe-results yet */
                    /*  then don't make any changes to the ospf interface config    */
                    if(not(probe-last-test-results)) {
                      var $status-msg = "event-script " _ $event-script-name _ " did not change
ospf interface " _ $int _ " metric because a complete RPM probe data set is not available.";
                      call log-status-msg($status-msg);
                    } else { /* the average delay info does exist  */
                      <area> {
```

```
                        <name> $int-area;
                        <interface> {
                          <name> $int;
                          <metric> $new-metric;
                        }
                        var $status-msg = "event-script " _ $event-script-name _ " changing ospf
interface " _ $int _ " metric from " _ $current-metric _ " to " _ $new-metric _ " based on rpm probe
data.";
                        call log-status-msg($status-msg);

                      }
                    }
                  }
                }
              }
            }
          }

          /* this is an empty change element used to determine if any changes need to be made */
          var $no-change := {
            <configuration> {
              <protocols> {
                <ospf> {
                  <area> {
                  }
                  <area> {
                  }
                  <area> {
                  }
                  <area> {
                  }
                }
              }
            }
          }

          /* compare the change element to the no-change element  */
          if ($change == $no-change){
            var $status-msg = "this message generated by " _ $event-script-name _ ": ospf probe
results show no change in link latency outside the allowed variance of " _ ($variance*100) _ "%." ;
            call log-status-msg($status-msg);
          } else {
            call config-change($change);
          }

}

/*************************************************************************************************/
/* this template returns the name of an ospf area given a logical ospf interface name*/
/*************************************************************************************************/
template ospf-int-area($int) {
          var $ospf-area = $configuration/protocols/ospf/area/interface[name == $int]/../name;
          expr $ospf-area;

}

/*************************************************************************************************
*********/
/* this template is assuming probe results in always come in us; it returns a proposed interface
metric    */
/* given the average delay info for an RPM probe test.                          */
/* if there is packet loss and $metric-up-on-loss = 1, then metric up value to 50000     */
```

```
/*************************************************************************************************
********/
template determine-metric($avg-delay, $difference) {
         /* let's say each ms for probe latency is worth a metric of 10 */
         if ($difference != 0 && $metric-up-on-loss == 1) {
           expr 50000;
         } else {
           var $new-metric = round((($avg-delay)div(1000))*10);
           expr $new-metric;
         }

}

/*************************************************************************************/
/* this template takes the config change element as input and commits the change via */
/* an exclusive commit, then verifies if the change was successful.  the template       */
/* returns a message indicating if the change was successful or not          */
/*************************************************************************************/
template config-change($change) {
         /* debug output */
         copy-of $change;

         /* make the configuration change */
         var $config-change-results := {
           call jcs:load-configuration($connection, $configuration = $change);
         }

         /* debug output */
         copy-of $config-change-results;

         if($config-change-results//self::xnm:error) {
           var $error-msg = "configuration change failed for reason " _ $config-change-results _ ";
could not configure RPM probes.  this msg generated by " _ $event-script-name;
           call log-status-msg($status-msg = $error-msg);
           <output> $error-msg;
           copy-of $change; /* for debug purposes */
         } else {
           var $success-msg = "configuration change to modify ospf metric values successful. this
msg generated by " _ $event-script-name;
           call log-status-msg($status-msg = $success-msg);
           <output> $success-msg;
           copy-of $change; /* for debug purposes */
         }
}

/*********************************************/
/* this template puts $status-msg in the log */
/*********************************************/
template log-status-msg($status-msg) {
         expr jcs:syslog(5, $status-msg);
}

/* EXPLANATION OF HOW THIS SOLUTION WORKS */

/* This automation solution sends a series of 15 RPM probes out over a defined testing period,
ranging from 60
* to 2592000 seconds, inclusive.  One time per testing period, the event-script checks the results of
the latest
* completed set of probe results and reads the average probe latency. If an interface probe's average
latency is
* greater than the allowed variance, then the script modifies that OSPF interface's metric. This
solution
* consists of two separate scripts – a commit script and an event script.
```

```
*
* In general, the commit script keeps the device configuration in line. The commit script performs
initial
* configuration of this solution (the user only need configure the commit and event scripts) and
then ensures
* that there is not an accidental configuration change that breaks it. The commit script does the
following upon
* each commit:
* 1)      Determines if the test-interval defined in the script lies within the valid range
60..2592000 seconds
* inclusive
* 2)      For each interface configured in OSPF with an apply-macro set to value 'monitor-link-
latency', the
* commit script checks to see if that interface has a /30 or /31 network mask. This solution only
runs
* on ospf interfaces with the apply-macro value 'monitor-link-latency'
* 3)      Determines the IP address for the remote side of the /30 or /31 OSPF link
* 4)      Configures an RPM probe with a target address of the link's remote side (probe name is
* ospf-metric-probe-<logical-interface>)
* 5)      If an interface is removed from OSPF, the commit script deletes that interface's solution
RPM probes
* from the configuration
* 6)      Configures a generated event (under the event-options hierarchy) that occurs at the
defined test
* interval once per testing period
* 7)      Configures an event-options policy that triggers upon the generated event and runs this
solution's event
* script
* 8)      Notifies the user at the CLI and in the logs of any changes it makes to the configuration
* 9)      If the commit script's $test-interval variable has been modified, the commit script
creates a new
* generated event with that interval, modifies the event-options policy to trigger upon that event,
and modifies
* the RPM probes for each OSPF interface to send an RPM probe at an interval close to (test-
interval)/15 seconds
*
*
* The commit script has three variables that can be modified by the user. The most important one is
the
* $test-interval variable. Modifying this value allows the user to change the length of the test
interval. This
* value is the length of the test interval, in seconds. Since 15 RPM probes for each OSPF interface
probe are
* sent in a given test interval, modifying this value causes the commit script to modify the probe-
interval for
* each probe whose name contains the regex 'ospf-metric-probe-'. Changing this value also causes a
new
* event-options generated event with the same value and a modification of the event-options policy
* 'monitor-ospf-interface-probes' to trigger on the corresponding generated event. In other words,
if the user
* changes this variable, the script will automatically make all the necessary changes to the
solution - the user
* need not make any further changes.
* The other two variables, $cmt-script-name and $event-script-name are the names of this commit
script and the
* name of the corresponding event script, respectively. The $event-script-name variable is used to
ensure that
* the event-options policy that runs the event script actually is configured for the correct event
script. The
* $cmt-script-name variable is the name of this commit script and is used to identify this script's
output in
* the system's logs.
*
```

```
* any actions taken by the commit script are logged and also noted at the CLI upon commit
* ---------------------------------------------------------------------------------------------
*
*The event-script portion of this solution is activated by an event-options policy configured by the
commit
* script. The event script performs the following tasks -
* 1)      Reads the RPM probe data only for the RPM probes whose names contain the regex ospf-metric-
probe- and
* finds the average probe latency for the most recently completed test
* 2)      Determines a proposed metric for each interface based on its most recent probe latency
(metric =
* (average delay in microsec)/1000 * 10, rounded to nearest integer.  In other words, 10*(average
delay in msec)
* rounded to nearest integer)
* 3)      Reads the current metric for each OSPF interface
* 4)      Determines if the new metric is within a specified variance of the existing metric.
*        a.  If variance is set to 40% (0.40),  if the new metric is within 40% of the existing
metric, the
* interface's metric is not changed
*        b.  Only if the new metric is more than 40% of the existing metric will that specific
interface's metric be
* changed
* 5)      Determines the area for each OSPF interface
* 6)      Changes the metric for each interface if the new metric is outside the allowed variance
* 7)      Each time the event script runs, it notifies the user via logs if it is making any changes
and what
* those changes are. If no changes are made, the script logs a message to that effect
*
*
* The event script has three variables that can be modified by the user, $variance,$event-script-
name, and
* $metric-up-on-loss.
*        a) $variance is a decimal value for the percentage of variance that an interface's RPM
probes can tolerate
* before the event script will modify its OSPF metric. The script is set to provide a metric of 10
for each ms
* of round trip latency. For example, when $variance = 0.4 (40%),  if an interface's metric is 500
and the most
* recent probe results show an average of 65,000 microseconds (us), the script will not modify the
OSPF metric
*  (65,000us = 65 ms = 650 metric; 30% variance). If a new set of results comes in showing a 75,000 us
average
*  (50% difference), then the script modifies the OSPF metric to 750.
*        b) $script-name is the name of this event-script and is used to identify this script's
messages in the
* system logs.
*        c) $metric-up-on-loss, when set to a value of 1, will metric up the appropriate OSPF
interface to 50,000
* if any of its RPM probes are lost on the most recent time-interval. If $metric-up-on-loss is set to
any
* value other than 1, then the interface metric is determined from the available RPM probe data for
that
* interface.
*
* any actions taken by the event-script are logged /*
```

## What to Do Next & Where to Go ...

*http://www.juniper.net/dayone*

The Day One book series is available for free download in PDF format. Select titles also feature a *Copy and Paste* edition for direct placement of Junos configurations. (The library is available in eBook format for iPads and iPhones from the Apple iBookstore, or download to Kindles, Androids, Blackberrys, Macs and PCs by visiting the Kindle Store. In addition, print copies are available for sale at Amazon or www.vervante.com.)

*http://www.juniper.net/books*

Check out the complete Juniper Networks Books library and the many book publishers that are participating, including *MPLS-Enabled Applications* by Ina Minei and Julian Lucek. This best-selling book on MPLS is in its third edition and has received five-star ratings.

*http://forums.juniper.net/jnet*

The Juniper-sponsored J-Net Communities forum is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions. Register to participate in this free forum.

*www.juniper.net/techpubs/*

Juniper Networks technical documentation includes everything you need to understand and configure all aspects of Junos, including MPLS. The documentation set is both comprehensive and thoroughly reviewed by Juniper engineering.

*www.juniper.net/training/fasttrack*

Take courses online, on location, or at one of the partner training centers around the world. The Juniper Network Technical Certification Program (JNTCP) allows you to earn certifications by demonstrating competence in configuration and troubleshooting of Juniper products. If you want the fast track to earning your certifications in enterprise routing, switching, or security use the available online courses, student guides, and lab guides.

### Junos Automation References

*http://www.juniper.net/dayone*

Visit the Day One page to follow the Junos Automation Series.

*http://www.juniper.net/us/en/training/elearning/junos_scripting.html*

A free online course covering Junos automation.

*http://www.juniper.net/us/en/community/junos/script-automation/#overview*

A Juniper script library with example scripts available at no charge.

*http://forums.juniper.net/t5/Junos-Automation-Scripting/bd-p/junos-automation*

The Junos automation online forum.