Multiprotocol Label Switching (MPLS)

JNCIS EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ **Define the functions and characteristics of RSVP**
- ✓ **Define the functions and characteristics of LDP**

In this chapter, we explore how the JUNOS software uses Multiprotocol Label Switching (MPLS) to forward user data traffic across a network. Before beginning this chapter, you should be familiar with the basic concepts of MPLS. You should know that a label-switched path (LSP) allows traffic forwarding based on label values encoded in a 4-octet shim header placed between the Layer 2 and Layer 3 headers. The routers along the path of an LSP fall into one of four categories—ingress, transit, penultimate, and egress. Each router performs a specific operation when user traffic is forwarded through the LSP. You should be familiar with the basic configuration and establishment of an LSP using the Resource Reservation Protocol (RSVP). Each interface requires support for MPLS packets, and the routing protocol daemon is informed of which operational interfaces support RSVP and MPLS. The LSP itself uses RSVP `Path` and `Resv` messages to establish the LSP. You can control the actual network links used by the LSP when you configure a named path, which supplies loose and strict hops. Finally, you should know that only BGP routes, by default, are allowed to use established LSPs but that specific IPv4 destinations can be associated with a particular LSP.

In this chapter, we examine the two signaling protocols used to establish LSPs—RSVP and the Label Distribution Protocol (LDP). In addition, we investigate how each protocol operates in an MPLS network, exchanges label information, and establishes label-switched paths.

Signaling Protocols

The JUNOS software supports two protocols for the dynamic signaling and setup of an LSP: RSVP and LDP. Each signaling option effectively establishes an LSP and forwards user data traffic using MPLS labels. However, each of the protocols accomplishes this in different ways. Let's see how each of the signaling mechanisms works.

Resource Reservation Protocol

The *Resource Reservation Protocol* (RSVP) was originally designed to provide end-user hosts with the ability to reserve network resources for data traffic flows. While this concept makes theoretical sense for a single enterprise network, it was never widely implemented for the Internet at large. In essence, the ISPs that make up the Internet didn't want individual customers altering the operation of their networks.

One of the basic concepts of RSVP is that a traffic flow consists of an identifiable session between two endpoints. Traditionally, these endpoints were the hosts in the network. The concept of a session ties neatly into the concept of an LSP, which transports a traffic flow between two individual routers in the network. This led network designers to extend the RSVP protocol specification to support traffic-engineering capabilities. This extended specification (RSVP-TE) allows an RSVP session to be established between two routers (or endpoints) in the network for the purpose of transporting a specific traffic flow.

Many of the original RSVP components are still used in an MPLS network, including the basic packet format of an RSVP message. Figure 7.1 displays the format of an RSVP message, which includes the following:

Version (4 bits) The version field displays the current version of RSVP represented by the message. A constant value of 0x1 is placed in this field.

Flags (4 bits) The Flags field is used to signal support for protocol extensions to neighboring RSVP routers. Currently only a single flag is defined for use in an MPLS network. The flag definitions are:

Bit 3 This flag bit is currently not defined and is set to a constant value of 0.

Bit 2 This flag bit is currently not defined and is set to a constant value of 0.

Bit 1 This flag bit is currently not defined and is set to a constant value of 0.

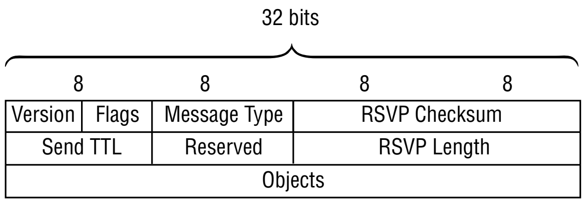
Bit 0 This flag bit signals support for the reduction of refresh overhead messages. This extension allows multiple RSVP messages to be bundled together into a single message format. When the bit is set to a value of 1, the local router supports the extensions. When the extensions are not supported, the flag is set to a value of 0.

Message Type (1 octet) This field displays the type of RSVP message encoded in the packet. The possible type codes are:

- 1—`Path`
- 2—`Resv`
- 3—`PathErr`
- 4—`ResvErr`
- 5—`PathTear`
- 6—`ResvTear`
- 7—`ResvConf`
- 12—`Bundle`
- 13—`Ack`
- 15—`Srefresh`
- 20—`Hello`
- 25—`Integrity Challenge`
- 26—`Integrity Response`

RSVP Checksum (2 octets) This field displays a standard IP checksum for the entire RSVP message. When the checksum is computed, the local router assumes this field contains all zeros.

FIGURE 7.1 RSVP message format



Send TTL (1 octet) This field contains the identical value used in the Time-to-Live (TTL) field of the IP header. It is used to detect a non-RSVP capable router along the path of an LSP.

Reserved (1 octet) This field is not used and must be set to a constant value of 0x00.

RSVP Length (2 octets) This field displays the length of the entire RSVP packet, including any optional objects that are attached to the message.

Objects (Variable) This variable-length field contains one or more RSVP objects. Each object is represented by a fixed-length header and a variable-length data field.

The information required to set up and maintain an RSVP session is encoded in multiple *objects* used in the various message types. Table 7.1 displays some RSVP objects and the messages that use them. The common reference for each object is a combination of its RSVP class name in addition to its specific object title. For example, IPv4/UDP is an object within the Session class. When you reference this object as a singular entity, it is called the IPv4/UDP Session object.

TABLE 7.1 RSVP Objects

Class Name	Class #	Object Name	C-Type #	Message Usage
Session	1	IPv4/UDP	1	`Path`
				`Resv`
				`PathErr`
				`ResvErr`
				`PathTear`
				`ResvTear`
		IPv6/UDP	2	`Path`
				`Resv`
				`PathErr`

TABLE 7.1 RSVP Objects *(continued)*

Class Name	Class #	Object Name	C-Type #	Message Usage
				`ResvErr`
				`PathTear`
				`ResvTear`
		LSP-Tunnel-IPv4	7	`Path`
				`Resv`
				`PathErr`
				`ResvErr`
				`PathTear`
				`ResvTear`
		LSP-Tunnel-IPv6	8	`Path`
				`Resv`
				`PathErr`
				`ResvErr`
				`PathTear`
				`ResvTear`
RSVP-Hop	3	IPv4	1	`Path`
				`Resv`
		IPv6	2	`Path`
				`Resv`
Integrity	4	Integrity	1	All types
Time-Values	5	Time-Values	1	`Path`
				`Resv`

TABLE 7.1 RSVP Objects *(continued)*

Class Name	Class #	Object Name	C-Type #	Message Usage
Error-Spec	6	IPv4	1	PathErr
				ResvErr
		IPv6	2	PathErr
				ResvErr
Scope	7	IPv4	1	Resv
		IPv6	2	Resv
Style	8	Style	1	Resv
Flowspec	9	Reserved	1	Resv
		Integrated Services	2	Resv
Filter-Spec	10	IPv4	1	Resv
		IPv6	2	Resv
		LSP-Tunnel-IPv4	7	Resv
		LSP-Tunnel-IPv6	8	Resv
Sender-Template	11	IPv4	1	Path
		IPv6	2	Path
		IPv6 Flow-Label	3	Path
		LSP-Tunnel-IPv4	7	Path
		LSP-Tunnel-IPv6	8	Path
Sender-Tspec	12	Integrated Services	2	Path
Adspec	13	Integrated Services	2	Path
Policy-Data	14	Type 1 Policy-Data	1	Path

TABLE 7.1 RSVP Objects *(continued)*

Class Name	Class #	Object Name	C-Type #	Message Usage
				`Resv`
Resv-Confirm	15	IPv4	1	`Resv`
		IPv6	2	`Resv`
Label	16	Label	1	`Resv`
		Generalized Label	2	`Resv`
		Waveband Switching	3	`Resv`
Label Request	19	Label Request (no label range)	1	`Path`
		Label Request (ATM label range)	2	`Path`
		Label Request (Frame Relay label range)	3	`Path`
		Generalized Label Request	4	`Path`
		Juniper Networks Extension	11	`Path`
Explicit Route	20	Explicit Route	1	`Path`
Record Route	21	Record Route	1	`Path`
				`Resv`
Hello	22	Hello Request	1	`Hello`
		Hello Acknowledgement	2	`Hello`
Message-ID	23	Message-ID	1	`Path`
				`Resv`

TABLE 7.1 RSVP Objects *(continued)*

Class Name	Class #	Object Name	C-Type #	Message Usage
Message-ID-Ack	24	Message-ID-Ack	1	`Path`
				`Resv`
		Message-ID-Nack	2	`Path`
Message-ID-List	25	Message-ID-List	1	`Path`
				`Resv`
		IPv4/Message-ID Src-List	2	`Path`
		IPv6/Message-ID Src-List	3	`Path`
		IPv4/Message-ID Mcast-List	4	`Path`
		IPv6/Message-ID Mcast-List	5	`Path`
Detour	63	Detour	7	`Path`
Challenge	64	Challenge	1	`Integrity Challenge`
				`Integrity Response`
Restart-Cap	131	Restart-Cap	1	`Hello`
Properties	204	Object-Type	1	`Path`
				`Resv`
Fast Reroute	205	Fast Reroute	1	`Path`
		Fast Reroute (existing implementations)	7	`Path`
Session Attribute	207	LSP-Tunnel-RA	1	`Path`
		LSP-Tunnel	7	`Path`

NOTE: A detailed explanation of every RSVP object listed is outside the scope of this book. Within this chapter, we'll focus only on the objects used to establish label-switched paths in an MPLS network.

Before we explore the format and use of RSVP objects, let's discuss the RSVP messages used in an MPLS network.

The *Path* Message

When an MPLS-capable router sets up an LSP, it generates a `Path` message and forwards it downstream to the egress router. The destination address of the message is the IP address of the egress router, but each device along the path examines the contents of the `Path` message. This hop-by-hop examination occurs since the ingress router sets the IP Router-Alert option in the message's IP header. The `Path` message may contain some of the following RSVP objects: Session, RSVP-Hop, Time-Values, Session Attribute, Sender Template, Sender-Tspec, Adspec, Explicit Route, Label Request, Properties, Record Route, Integrity, Fast Reroute, and Detour.

FIGURE 7.2 Sample MPLS network

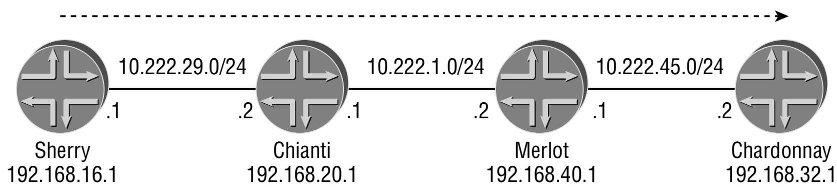


Figure 7.2 shows a simple MPLS network consisting of four routers; Sherry, Chianti, Merlot, and Chardonnay. An MPLS LSP is established from Sherry to Chardonnay using RSVP. We use the `show mpls lsp ingress` command on the Sherry router to verify that the LSP is operational:

```
user@Sherry> show mpls lsp ingress
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0                  *     Sherry-to-Char
Total 1 displayed, Up 1, Down 0
```

The configuration on Sherry that created the ***Sherry-to-Char*** LSP looks like this:

```
user@Sherry> show configuration protocols mpls
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
}
interface all;
```

When the configuration was committed, Sherry generated an RSVP `Path` message addressed to the egress address of 192.168.36.1 and sent it downstream. After some processing at the Chianti router, the `Path` message is advertised further downstream to Merlot. We see the message arrive at Merlot and appear in the output of a `traceoptions` file, which is collecting RSVP information:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

When Merlot receives this particular `Path` message, it examines the information in the `Session7` and `Sender7` objects to determine if this unique combination was already seen in a previous `Path` message. Because this is a new LSP, we know that the message has not been seen by Merlot. This "newness" means that Merlot creates a new RSVP soft-state set of information known as a *Path State Block*. This soft-state information includes information from the `Path` message such as the Session, Sender Template, and Sender-Tspec objects. Additionally, the interface address of the previous hop along the LSP (10.222.1.1) is stored, which allows the router to send a `Resv` message to the appropriate upstream device.

Because Merlot is not the egress router, it then prepares to send the `Path` message further downstream. Before doing so, it adds the outgoing interface address (10.222.45.1) to the Record Route object and places it in the RSVP-Hop object. We can see the message sent to Chardonnay in the `traceoptions` file as

```
Apr 24 10:56:24 RSVP send Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/2.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.45.1/0x08550264
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 12  10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 28  10.222.45.1 10.222.1.1 10.222.29.1
```

After the Path message reaches the egress router of Chardonnay, the resources of the LSP are established in the network by Resv messages.

The *Resv* Message

The Chardonnay router, our LSP egress router, generates a Resv message and forwards it upstream to Merlot. The destination address of the Resv message is the interface address of the Merlot router. This information is gleaned from the RSVP-Hop object in the Path message received by Chardonnay. An individual Resv message may contain some of the following RSVP objects: Session, RSVP-Hop, Time-Values, Style, Flowspec, Filter-Spec, Label, Record Route, and Integrity.

The first Resv message received on Merlot is from the Chardonnay router. We see the details of this message in the output of our traceoptions file:

```
Apr 24 10:56:24 RSVP recv Resv 10.222.45.2->10.222.45.1 Len=120 so-0/1/2.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.45.2/0x08550264
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  3
Apr 24 10:56:24   RecRoute Len 12  10.222.45.2
```

When Merlot receives this first Resv message, it creates an additional set of RSVP soft state information known as a *Resv State Block*. This state information is uniquely identified by the data in the Session, RSVP-Hop, and Flowspec objects. The Resv State Block stores these data fields as well as the outgoing interface for the traffic flow and the style of the particular reservation request. Merlot then consults the Path State Block information previously stored and locates the next upstream router. This lookup is keyed against the information stored in the Session and Filter-Spec objects.

When a matching Path State Block is found, Merlot records the label advertised from the downstream router. In our particular example, Chardonnay advertised a label value of 3, signaling Merlot to perform penultimate hop popping (PHP). Because Merlot is not the ingress router, a new Resv message is formulated for advertisement upstream to Chianti. Before sending the message, Merlot adds its outgoing interface address (10.222.1.2) to the Record Route object and places it in the RSVP-Hop object. Additionally, a label value of 100,001 is allocated and included in the Label object of the Resv message. We can see these message details in the traceoptions file on Merlot:

```
Apr 24 10:56:24 RSVP send Resv 10.222.1.2->10.222.1.1 Len=128 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.2/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  100001
Apr 24 10:56:24   RecRoute Len 20  10.222.1.2 10.222.45.2
```

When the flow of Resv messages reaches the Sherry router, the LSP is successful established.

The *PathErr* Message

An RSVP PathErr message is sent hop by hop to the ingress router from the device noticing the error. PathErr messages do not destroy any established soft state in the network, but are used simply to signal error information, often a processing problem with a Path message, to the ingress router. These messages are sent in a hop-by-hop fashion upstream and are addressed to the interface address of the previous hop. A PathErr message may contain some of the following RSVP objects: Session, Error-Spec, Sender-Template, and Sender-Tspec.

Using Figure 7.2 as a guide, we once again build an LSP from Sherry to Chardonnay. The LSP is provided an explicit route through the network using a named path of ***strict-hops***, which is applied to the LSP as so:

```
user@Sherry> show configuration protocols mpls
label-switched-path Sherry-to-Chardonnay {
    to 192.168.32.1;
    bandwidth 15m;
    no-cspf;
    primary strict-hops;
}
path strict-hops {
    10.222.29.2 strict;
    10.222.1.2 strict;
    10.222.111.2 strict;
}
interface all;
```

The final address listed in the ***strict-hops*** path is 10.222.111.2, which does not correspond to the address of the Chardonnay router. This causes the Path message processing to fail at Merlot, since the next hop in the explicit route is not reachable. Merlot generates a PathErr message address to 10.222.1.1, the interface address of Chianti, to signal the ingress router of the problem. The message details are visible in the traceoptions file:

```
Apr 25 07:06:00 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 25 07:06:00   Session7 Len 16 192.168.32.1(port/tunnel ID 44222) Proto 0
Apr 25 07:06:00   Hop      Len 12 10.222.1.1/0x084ec198
Apr 25 07:06:00   Time     Len  8 30000 ms
Apr 25 07:06:00   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 25 07:06:00   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 25 07:06:00   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 25 07:06:00   ADspec   Len 48
Apr 25 07:06:00   SrcRoute Len 20  10.222.1.2 S 10.222.111.2 S
Apr 25 07:06:00   LabelRequest Len  8 EtherType 0x800
Apr 25 07:06:00   Properties Len 12 Primary path
Apr 25 07:06:00   RecRoute Len 20  10.222.1.1 10.222.29.1

Apr 25 07:06:00 RSVP originate PathErr 10.222.1.2->10.222.1.1
Apr 25 07:06:00   Explicit Route: bad strict route LSP Sherry-to-Char(1/44222)
Apr 25 07:06:00 RSVP send PathErr 10.222.1.2->10.222.1.1 Len=84 so-0/1/0.0
Apr 25 07:06:00   Session7 Len 16 192.168.32.1(port/tunnel ID 44222) Proto 0
Apr 25 07:06:00   Error    Len 12 code 24 value 2 flag 0 by 10.222.1.2
Apr 25 07:06:00   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 25 07:06:00   Tspec    Len 36 rate 15Mbps size 15Mbps peak Infbps m 20 M 1500
```

From Merlot's perspective, we see the arrival of the Path message from Sherry, which contains an explicit route shown as SrcRoute Len 20 10.222.1.2 S 10.222.111.2 S. As Merlot is not capable of forwarding the Path message to 10.222.111.2, it generates a PathErr message containing the error information of Explicit Route: bad source route. This message travels upstream to the Sherry router, where the status of the LSP is listed as nonoperational. We can gather valuable information as to why the LSP is not working from the output of the show mpls lsp ingress extensive command:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Dn, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: (none)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary   strict-hops      State: Dn
    Bandwidth: 15Mbps
    2 Apr 25 07:06:33  10.222.1.2: Explicit Route: bad strict route[3 times]
    1 Apr 25 07:05:53  Originate Call
  Created: Fri Arp 25 07:05:47 2003
Total 1 displayed, Up 0, Down 1
```

The *ResvErr* Message

An RSVP ResvErr message is also used to alert routers in the network to potential problems. The message is sent hop by hop to the egress router from the device noticing the error. Like the PathErr message, ResvErr messages do not destroy any established soft state in the network.

The messages are addressed to the IP interface address of the next downstream hop along the path. A `ResvErr` message may contain some of the following RSVP objects: Session, RSVP-Hop, Error-Spec, Style, Flowspec, and Filter-Spec.

Once the ***Sherry-to-Char*** LSP is established in Figure 7.2, the interface address of the Chianti router is changed from 10.222.29.2 /24 to 10.222.29.100 /24. This address change causes Chianti to include 10.222.29.100 in the RSVP-Hop object in its next `Resv` refresh message to Sherry. We see this message in the `traceoptions` output on Chianti:

```
Apr 25 10:35:37 RSVP send Resv 10.222.29.100->10.222.29.1 Len=136 ge-0/2/0.0
Apr 25 10:35:37   Session7 Len 16 192.168.32.1(port/tunnel ID 44234) Proto 0
Apr 25 10:35:37   Hop      Len 12 10.222.29.100/0x084fb0cc
Apr 25 10:35:37   Time     Len  8 30000 ms
Apr 25 10:35:37   Style    Len  8 FF
Apr 25 10:35:37   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 25 10:35:37   Filter7  Len 12 192.168.61.1(port/lsp ID  1)
Apr 25 10:35:37   Label    Len  8  100007
Apr 25 10:35:37   RecRoute Len 28  10.222.29.100 10.222.1.2 10.222.45.2
```

When Sherry receives this `Resv` message, it compares its contents against the current Resv State Block and finds that the interface address has been changed. Sherry then generates a `ResvErr` message and forwards it downstream to the egress router. This message is received by Chianti, which then forwards it along to Merlot:

```
Apr 25 10:35:37 RSVP recv ResvErr 10.222.29.1->10.222.29.100 Len=104 ge-0/2/0.0
Apr 25 10:35:37   Session7 Len 16 192.168.32.1(port/tunnel ID 44234) Proto 0
Apr 25 10:35:37   Hop      Len 12 10.222.29.1/0x084fb0cc
Apr 25 10:35:37   Error    Len 12 code 4 value 0 flag 0 by 10.222.29.1
Apr 25 10:35:37   Style    Len  8 FF
Apr 25 10:35:37   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 25 10:35:37   Filter7  Len 12 192.168.16.1(port/lsp ID  1)

Apr 25 10:35:37 RSVP send ResvErr 10.222.1.1->10.222.1.2 Len=104 so-0/1/0.0
Apr 25 10:35:37   Session7 Len 16 192.168.32.1(port/tunnel ID 44234) Proto 0
Apr 25 10:35:37   Hop      Len 12 10.222.1.1/0x084ec198
Apr 25 10:35:37   Error    Len 12 code 4 value 0 flag 0 by 10.222.29.1
Apr 25 10:35:37   Style    Len  8 FF
Apr 25 10:35:37   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 25 10:35:37   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
```

The *PathTear* Message

Once an LSP is established in the network, RSVP maintains its soft-state data structures as long as the LSP is operational. Once the LSP is no longer needed, the state information is removed from the router's memory. This removal of state information is the function of the PathTear message. The message travels downstream to the egress router, removing the Path State Block and Resv State Block information along the way. The destination address of the PathTear message is the IP address of the egress router, but the IP Router-Alert option is set to allow each device along the path to examine the message contents. The PathTear message may contain some of the following RSVP objects: Session, RSVP-Hop, Sender-Template, and Sender-Tspec.

The ***Sherry-to-Char*** LSP from Figure 7.2 is operational in the network. We verify its status on the ingress router of Sherry:

```
user@Sherry> show mpls lsp
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0                  *     Sherry-to-Char
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
```

Suppose the network link between the Chianti and Merlot routers now fails. This link failure causes Merlot to generate a PathTear message addressed to the egress address of 192.168.32.1. The message replicates the information originally found in the Path message that established the LSP. This allows Merlot and Chardonnay to remove all RSVP state information associated with the ***Sherry-to-Char*** LSP. We can view the contents of the message in the output of the traceoptions file on Merlot:

```
Apr 25 09:56:54 RSVP send PathTear 192.168.16.1->192.168.32.1 Len=84 so-0/1/2.0
Apr 25 09:56:54   Session7 Len 16 192.168.32.1(port/tunnel ID 44230) Proto 0
Apr 25 09:56:54   Hop      Len 12 10.222.45.1/0x08550264
Apr 25 09:56:54   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 25 09:56:54   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
```

The *ResvTear* Message

In a similar fashion to the PathTear message, a ResvTear message also removes RSVP state information in the network. The ResvTear message travels upstream to the ingress router in a hop-by-hop fashion. The destination address of the message is the IP interface address of the previous hop along the path. The ResvTear message may contain some of the following RSVP objects: Session, RSVP-Hop, Style, and Filter-Spec.

When the network link between the Chianti and Merlot routers fails, the state information on Chianti and Sherry is no longer needed. The Chianti router generates a ResvTear message addressed to the interface address of Sherry—10.222.29.1. The message replicates the information originally found in the `Resv` message that established the LSP. We can view the output of the `traceoptions` file on Chianti and see the `ResvTear` message sent upstream:

```
Apr 25 09:54:57 RSVP send ResvTear 10.222.29.2->10.222.29.1 Len=56 ge-0/2/0.0
Apr 25 09:54:57   Session7 Len 16 192.168.32.1(port/tunnel ID 44230) Proto 0
Apr 25 09:54:57   Hop      Len 12 10.222.29.2/0x084fb0cc
Apr 25 09:54:57   Style    Len  8 FF
Apr 25 09:54:57   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
```

RSVP Objects

Now that we have some understanding of how RSVP uses its messages to establish and withdraw its soft-state information, we can focus on the details of the objects used in those messages. It should be clear at this point in the chapter that the information carried in each message is the key to understanding the operation of an RSVP-based MPLS network.

The LSP-Tunnel-IPv4 Session Object

The *LSP-Tunnel-IPv4 Session object* is perhaps the most widely seen object in an MPLS network. Not only is the object required in every RSVP message type, but it's also the key to uniquely identifying an LSP and its soft-state information. Figure 7.3 shows the format of the Session object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the LSP-Tunnel-IPv4 Session object, a constant value of 16 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Session object falls within the Session class, which uses a value of 1.

The values assigned to the RSVP classes are encoded using one of three bit patterns. These patterns are used when the received class value is not recognized, or unknown. The bit patterns (where b is a 0 or 1) are:

0bbbbbbb The receipt of an unknown class value using this bit pattern causes the local router to reject the message and return an error to the originating node.

10bbbbbb The receipt of an unknown class value using this bit pattern causes the local router to ignore the object. It is not forwarded to any RSVP neighbors, and no error messages are generated.

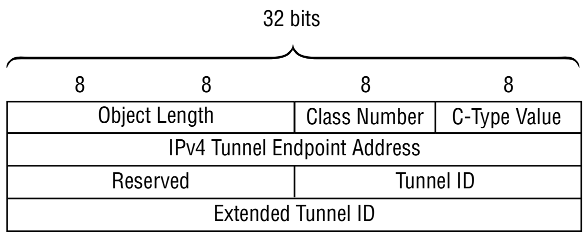
11bbbbbb The receipt of an unknown class value using this bit pattern causes the local router to also ignore the object. In this case, however, the router forwards the object to its neighbors without examining or modifying it in any way.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The LSP-Tunnel-IPv4 Session object uses a value of 7 for its C-Type.

IPv4 Tunnel Endpoint Address (4 octets) This field displays the IPv4 address of the LSP egress router.

Reserved (2 octets) This field is not used and must be set to a constant value of 0x0000.

FIGURE 7.3 The LSP-Tunnel-IPv4 Session object



Tunnel ID (2 octets) This field contains a unique value generated by the ingress router of the LSP. This helps to distinguish the particular LSP from other paths originating at the same ingress router. The selected value remains constant throughout the life of the LSP.

Extended Tunnel ID (4 octets) An ingress router places its IPv4 address in this field to further identify the session. This allows the LSP to be uniquely identified by the ingress-egress router addresses. Although the use of this field is not required, the JUNOS software places the ingress address here by default.

We see the Session object appear in a `Path` message received by the Merlot router in Figure 7.2:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

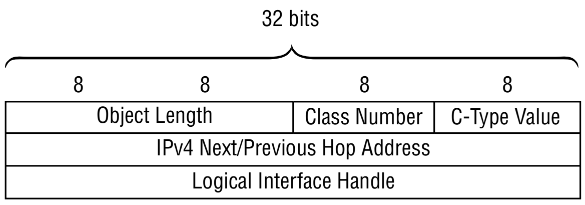
The C-Type value of 7 and the Session class of the object appears in the output as `Session7`. Within the object itself, we see the egress router address of 192.168.32.1 displayed along with the tunnel ID value of 44214.

The IPv4 RSVP-Hop Object

The *IPv4 RSVP-Hop object* is used to identify the IP interface address of the neighboring RSVP router. It allows each device in the network to store information in both the Path and Resv State Blocks. In addition, it is instrumental in properly routing RSVP messages in the network. Figure 7.4 shows the format of the RSVP-Hop object, which contains the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the IPv4 RSVP-Hop object, a constant value of 12 is placed in this field.

FIGURE 7.4 The IPv4 RSVP-Hop object



Class Number (1 octet) The value that represents the object's class is placed into this field. The RSVP-Hop object falls within the RSVP-Hop class, which uses a value of 3.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The IPv4 RSVP-Hop object uses a value of 1 for its C-Type.

IPv4 Next/Previous Hop Address (4 octets) This field displays the IPv4 address of the neighboring RSVP router.

Logical Interface Handle (4 octets) Each interface that transmits and receives RSVP messages is assigned a unique 32-bit value known as the *logical interface handle*. The sending router populates this field, which contains the unique ID value. This value allows the router receiving the message to associate it with the appropriate logical interface.

The RSVP-Hop object also appears in `Path` messages received by the Merlot router for the ***Sherry-to-Char*** LSP:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

When Merlot receives the `Path` message, it associates the 10.222.1.1 address as the interface address of the next upstream router. The logical interface handle of 0x084ec198 is stored in the Path State Block for this session. As `Resv` messages arrive from the downstream router, Merlot uses the information in the RSVP-Hop object to forward its own `Resv` message upstream, with the address retrieved from the object encoded as the destination address. Additionally, Merlot places the logical interface handle it received in the `Path` message into the RSVP-Hop object it places in its own Resv message. The inclusion of the logical handle allows the upstream router to correlate the received Resv message with the correct interface.

The Time-Values Object

The *Time-Values object* is included in all Path and Resv messages used in the MPLS network. It contains a refresh value that is used by the receiving router to calculate the lifetime of the RSVP soft-state information. The value advertised in the Time-Values object is used by the local router to regenerate the appropriate message and send it to its RSVP neighbor. Figure 7.5 displays the format of the Time-Values object, which includes these fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the Time-Values object, a constant value of 8 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Time-Values object falls within the Time-Values class, which uses a value of 5.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Time-Values object uses a value of 1 for its C-Type.

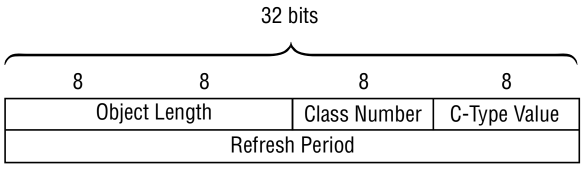
Refresh Period (4 octets) This field displays the refresh time, in milliseconds, of the particular RSVP message. The JUNOS software uses a default value of 30,000 (30 seconds) for the refresh period.

The Merlot router in Figure 7.2 is receiving Resv messages for the ***Sherry-to-Char*** LSP from its downstream neighbor. The output of a traceoptions file shows the details of that Resv message:

```
Apr 24 10:56:24 RSVP recv Resv 10.222.45.2->10.222.45.1 Len=120 so-0/1/2.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.45.2/0x08550264
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  3
Apr 24 10:56:24   RecRoute Len 12  10.222.45.2
```

The 30000 ms refresh time advertised in the Resv message informs Merlot that it should expect a refresh of this RSVP state every 30 seconds. In addition, the advertised value also allows Merlot to calculate the timeout value for the state information.

FIGURE 7.5 The Time-Values object



The IPv4 Error-Spec Object

The *IPv4 Error-Spec object* appears in either a `PathErr` or `ResvErr` message to provide a root cause for the error message. The object includes the IP address of the router that first discovered the error, which allows all devices in the network to locate the failure point. The fields of the IPv4 Error-Spec object, shown in Figure 7.6, include:

Object Length (2 octets) This field displays the total length of the RSVP object. For the IPv4 Error-Spec object, a constant value of 12 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Error-Spec object falls within the Error-Spec class, which uses a value of 6.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The IPv4 Error-Spec object uses a value of 1 for its C-Type.

IPv4 Error Node Address (4 octets) This field displays the IPv4 address of the RSVP router that detected an error condition.

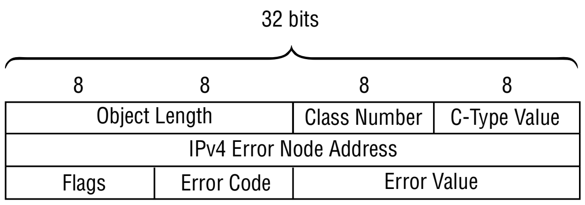
Flags (1 octet) This field contains flags that allow certain information to be carried in the Error-Spec object. Currently, two flags are defined for use within RSVP. Bit 0 (0x01) is used only when the object is contained in a `ResvErr` message. It signifies that an active reservation was in place and remains in place on the router detecting the device. Bit 1 (0x02) is also used only when the object is contained in a `ResvErr` message. It signifies that a reservation failure occurred that required more resources than the recipient of the message requested.

Error Code (1 octet) This field displays the major error encountered by the RSVP router.

Error Value (2 octets) This field displays additional specific information pertaining to the reported error code.

NOTE For a complete list of error codes and values, please refer to Request for Comments (RFC) 2205, "Resource Reservation Protocol—Version 1 Functional Specification."

FIGURE 7.6 The IPv4 Error-Spec object



After the ***Sherry-to-Char*** LSP is established in Figure 7.2, the IP interface address of the Chianti router is changed. This causes Sherry (10.222.29.1) to generate a `ResvErr` message containing the IPv4 Error-Spec object. The contents of this message are seen on the Merlot router:

```
Apr 25 10:35:19 RSVP recv ResvErr 10.222.1.1->10.222.1.2 Len=104 so-0/1/0.0
Apr 25 10:35:19   Session7 Len 16 192.168.32.1(port/tunnel ID 44234) Proto 0
Apr 25 10:35:19   Hop      Len 12 10.222.1.1/0x084ec198
Apr 25 10:35:19   Error    Len 12 code 4 value 0 flag 0 by 10.222.29.1
Apr 25 10:35:19   Style    Len  8 FF
Apr 25 10:35:19   Flow     Len 36 rate 100Mbps size 100Mbps peak Infbps m 20 M
1500
Apr 25 10:35:19   Filter7  Len 12 192.168.61.1(port/lsp ID  1)
```

The Style Object

The *Style object* is used in a `Resv` message to determine how reservations are made in the network. By default, the JUNOS software creates a unique reservation for each RSVP session and sender. This means that every new combination of Session and Sender-Template objects requires a new reservation of resources. This reservation style is known as *fixed filter* (FF). A second reservation style called *shared explicit* (SE) allows a single set of reserved resources for a session to be shared among multiple senders. In other words, multiple Sender-Template objects that share a common Session object are allowed to use the same resources. Finally, RSVP devices may use a reservation style known as *wildcard filter* (WF). This allows for a set of resources to be shared among all possible senders. The wildcard filter reservation style is not supported by the JUNOS software.

Figure 7.7 shows the format of the Style object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the Style object, a constant value of 8 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Style object falls within the Style class, which uses a value of 8.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Style object uses a value of 1 for its C-Type.

Flags (1 octet) This field contains bit flags used to relay information to RSVP routers. No flags are currently defined.

Option Vector (3 octets) This field displays the type of reservation being established by the object. The field is divided into three distinct portions. The first portion of the field uses the 19 most significant bits and is a reserved value with all of the bits set to 0.

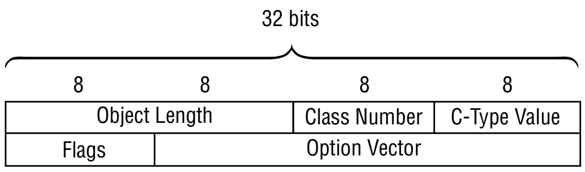
The second portion of the field uses the next two most significant bits (3 and 4) to specify whether the reservation is shared. A value of 01 is a distinct reservation and is used by the FF style. A value of 10 represents a shared reservation and is used by the WF and SE styles.

The final 3 bits of the field (0, 1, and 2) determines which senders may use the reservation. A value of 001 is a wildcard notation that allows any possible sender to utilize the resources. This is used by the WF reservation style. A value of 010 denotes an explicit use of resources by only known senders. This explicit notation is used by the FF and SE reservation styles.

The final two portions of the field (5 total bits) are combined to yield the three distinct reservation styles as follows:

- 01010—Fixed filter (FF)
- 10001—Wildcard filter (WF)
- 10010—Shared explicit (SE)

FIGURE 7.7 The Style object



Refer back to Figure 7.2 and the LSP established between Sherry and Chardonnay. When the Merlot router transmits a `Resv` message upstream for that session, it details the specific style for the reservation in the Style object:

```
Apr 24 10:56:24 RSVP send Resv 10.222.1.2->10.222.1.1 Len=128 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.2/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  100001
Apr 24 10:56:24   RecRoute Len 20  10.222.1.2 10.222.45.2
```

Using the output from the `traceoptions` file on Merlot, we see that the ***Sherry-to-Char*** LSP is using the default JUNOS software reservation style of fixed filter.

The Integrated Services Flowspec Object

The *Integrated Services Flowspec object* appears in `Resv` messages and contains information pertaining to the bandwidth request of the LSP, if any. The format and use of the object fields was designed for a controlled load environment where average and peak data rates could be defined. Modern implementations, including the JUNOS software, don't use these fields as they were originally intended. Instead, the bandwidth reservation for the LSP is placed there. Figure 7.8 details the format of the Integrated Services (IntServ) Flowspec object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the Intserv Flowspec object, a constant value of 36 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Intserv Flowspec object falls within the Flowspec class, which uses a value of 9.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Intserv Flowspec object uses a value of 2 for its C-Type.

Version Number (2 octets) This 2-octet field uses the first 4 bits to encode the version of the object message format. A constant value of 0 is placed in this portion of the field. The remaining 20 bits of the field are reserved and must be set to a constant value of 0x00000.

Flowspec Length (2 octets) This field displays the total number of 32-bit words that make up the Service portion of the object. When used in an MPLS environment, this includes all of the remaining fields in the figure. A constant value of 7 is placed in this field.

Service Number (1 octet) This field displays the type of Integrated Service this Flowspec object supports. A constant value of 5 is placed in this field to represent a controlled load session.

Reserved (1 octet) This field is not used and is set to a constant value of 0x00.

Data Length (2 octets) This field displays the total number of 32-bit words that make up the data portion of the object. As with the Flowspec length field, this includes all of the remaining fields in the figure. A constant value of 6 is placed in this field.

Parameter ID (1 octet) This field displays the ID value associated with the controlled load data. For the IntServ Flowspec object, a value of 127 is placed in this field.

Parameter Flags (1 octet) This field contains bit flags used to advertise information to other routers concerning the object. No flag values are currently defined, and this field is set to a constant value of 0x00.

Parameter Length (2 octets) This field displays the total number of 32-bit words that make up the parameter portion of the object. As before, this includes all of the remaining fields in the figure. A constant value of 5 is placed in this field.

Token Bucket Rate (4 octets) This field displays the bandwidth reservation request for the LSP, if one was configured. A value of 0x00000000 in this field represents a reservation with no bandwidth requirement.

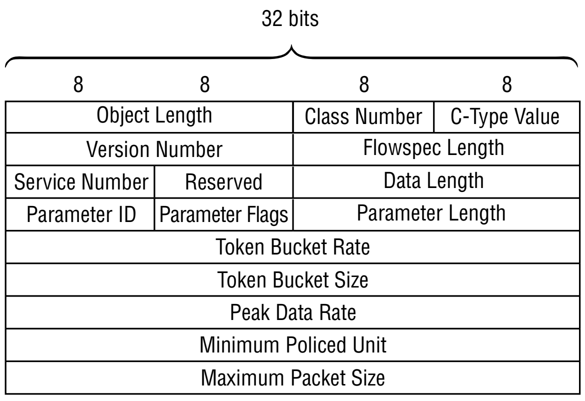
Token Bucket Size (4 octets) This field also displays the bandwidth reservation request for the LSP. Again, a value of 0x00000000 means that no bandwidth was requested.

Peak Data Rate (4 octets) This field was originally designed to display the maximum load of data able to be sent through the reservation. This field is not used by the JUNOS software, and a constant value of 0x7f800000 is placed in this field, which represents an infinite peak bandwidth limit.

Minimum Policed Unit (4 octets) This field displays the smallest packet size supported through the LSP. The router treats all packets smaller than 20 bytes as if they were 20 bytes in length.

Maximum Packet Size (4 octets) This field displays the largest packet size supported through the LSP. The router treats all packets larger than 1500 bytes as if they were 1500 bytes in length.

FIGURE 7.8 The Integrated Services Flowspec object



Using the ***Sherry-to-Char*** LSP in Figure 7.2 as a guide, the IntServ Flowspec object is carried in the `Resv` messages that sets up and maintains the LSP. The Merlot router is sending a `Resv` message upstream to Chianti, whose contents are visible in a `traceoptions` file:

```
Apr 24 10:56:24 RSVP send Resv 10.222.1.2->10.222.1.1 Len=128 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.2/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  100001
Apr 24 10:56:24   RecRoute Len 20  10.222.1.2 10.222.45.2
```

Within the object, we see that no bandwidth requests are configured for this LSP. This is evident by the `rate 0bps` and `size 0bps` output on the router. The infinite peak data rate associated with the LSP is represented as `peak Infbps`, while the minimum (`m`) and maximum (`M`) packet sizes are also displayed.

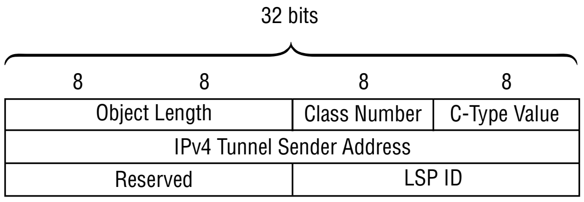
The LSP-Tunnel-IPv4 Filter-Spec Object

The *LSP-Tunnel-IPv4 Filter-Spec object* is contained in a `Resv` message and is used to uniquely identify the sender for the LSP. This information is useful for sharing resources in the network for a single RSVP session with multiple senders. The shared explicit reservation style requires this information in the Resv State Block to adequately allocate the network resources. Figure 7.9 shows the format of the Filter-Spec object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the LSP-Tunnel-IPv4 Filter-Spec object, a constant value of 12 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Filter-Spec object falls within the Filter-Spec class, which uses a value of 10.

FIGURE 7.9 The LSP-Tunnel-IPv4 Filter-Spec object



C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The LSP-Tunnel-IPv4 Filter-Spec object uses a value of 7 for its C-Type.

IPv4 Tunnel Sender Address (4 octets) This field displays the IPv4 address of the LSP ingress router.

Reserved (2 octets) This field is not used and must be set to a constant value of 0x0000.

LSP ID (2 octets) This field contains a unique value generated by the ingress router of the LSP. This helps to distinguish the particular sender and its resources from other paths originating at the same ingress router for the same RSVP session.

We can see the Filter-Spec object in a `Resv` message received by the Merlot router in Figure 7.2:

```
Apr 24 10:56:24 RSVP recv Resv 10.222.45.2->10.222.45.1 Len=120 so-0/1/2.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.45.2/0x08550264
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  3
Apr 24 10:56:24   RecRoute Len 12  10.222.45.2
```

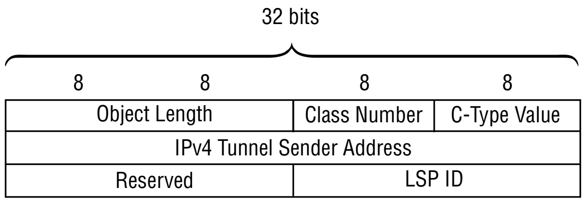
The C-Type value of 7 and the Filter-Spec class of the object appear in the output as `Filter7`. Within the object itself, we see the ingress router address of 192.168.16.1 displayed along with the LSP ID value of 1.

The LSP-Tunnel-IPv4 Sender Template Object

The *LSP-Tunnel-IPv4 Sender Template object* is contained in a `Path` message and is used to uniquely identify the sender for the LSP. This information contained in this object is identical to that found in the LSP-Tunnel-IPv4 Filter-Spec object. As such, it is useful for differentiating among sending sources, or LSP IDs, within a single RSVP session. This information is stored in the Path State Block for each router along the path of the LSP. Figure 7.10 displays the format of the Sender Template object, which includes these fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the LSP-Tunnel-IPv4 Sender Template object, a constant value of 12 is placed in this field.

FIGURE 7.10 The LSP-Tunnel-IPv4 Sender Template object



Class Number (1 octet) The value that represents the object's class is placed into this field. The Sender Template object falls within the Sender Template class, which uses a value of 11.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The LSP-Tunnel-IPv4 Sender Template object uses a value of 7 for its C-Type.

IPv4 Tunnel Sender Address (4 octets) This field displays the IPv4 address of the LSP ingress router.

Reserved (2 octets) This field is not used and must be set to a constant value of 0x0000.

LSP ID (2 octets) This field contains a unique value generated by the ingress router of the LSP. This helps to distinguish the particular sender and its resources from other paths originating at the same ingress router for the same RSVP session.

Using Figure 7.2 as a guide, we can view a `Path` message arriving on the Merlot router for the ***`Sherry-to-Char`*** LSP:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

The C-Type value of 7 and the Sender Template class of the object appear in the output as `Sender7`. Within the object itself, we see the ingress router address of 192.168.16.1 displayed along with the LSP ID value of 1.

The Integrated Services Sender-Tspec Object

The *Integrated Services Sender-Tspec object* appears in `Path` messages and contains information pertaining to the bandwidth request of the LSP, if any. Like the IntServ Flowspec object, the format and use of the object fields was designed for a controlled load environment where average and peak data rates could be defined. Modern implementations, including the JUNOS software, place the bandwidth reservation for the LSP in these fields. Figure 7.11 details the format of the Integrated Services (IntServ) Sender-Tspec object, which includes the following:

Object Length (2 octets) This field displays the total length of the RSVP object. For the Intserv Sender-Tspec object, a constant value of 36 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Intserv Sender-Tspec object falls within the Sendter-Tspec class, which uses a value of 12.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Intserv Sender-Tspec object uses a value of 2 for its C-Type.

Version Number (2 octets) This 2-octet field uses the first 4 bits to encode the version of the object message format. A constant value of 0 is placed in this portion of the field. The remaining 20 bits of the field are reserved and must be set to a constant value of 0x00000.

Tspec Length (2 octets) This field displays the total number of 32-bit words that make up the Service portion of the object. When used in an MPLS environment, this includes all of the remaining fields in the figure. A constant value of 7 is placed in this field.

Service Number (1 octet) This field displays the type of Integrated Service this Sender-Tspec Object supports. A constant value of 1 is placed in this field to represent a default controlled load session.

Reserved (1 octet) This field is not used and is set to a constant value of 0x00.

Data Length (2 octets) This field displays the total number of 32-bit words that make up the data portion of the object. As with the Tspec Length field, this includes all of the remaining fields in the figure. A constant value of 6 is placed in this field.

Parameter ID (1 octet) This field displays the ID value associated with the controlled load data. For the IntServ Sender-Tspec object, a value of 127 is placed in this field.

Parameter Flags (1 octet) This field contains bit flags used to advertise information to other routers concerning the object. No flag values are currently defined, and this field is set to a constant value of 0x00.

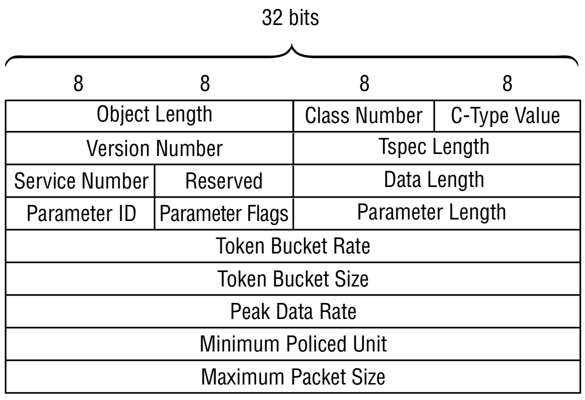
Parameter Length (2 octets) This field displays the total number of 32-bit words that make up the parameter portion of the object. As before, this includes all of the remaining fields in the figure. A constant value of 5 is placed in this field.

Token Bucket Rate (4 octets) This field displays the bandwidth reservation request for the LSP, if one was configured. A value of 0x00000000 in this field represents a reservation with no bandwidth requirement.

Token Bucket Size (4 octets) This field also displays the bandwidth reservation request for the LSP. Again, a value of 0x00000000 means that no bandwidth was requested.

Peak Data Rate (4 octets) This field was originally designed to display the maximum load of data able to be sent through the reservation. This field is not used by the JUNOS software, and a constant value of 0x7f800000 is placed in this field, which represents an infinite peak bandwidth limit.

FIGURE 7.11 The Integrated Services Sender-Tspec object



Minimum Policed Unit (4 octets) This field displays the smallest packet size supported through the LSP. The router treats all packets smaller than 20 bytes as if they were 20 bytes in length.

Maximum Packet Size (4 octets) This field displays the largest packet size supported through the LSP. The router treats all packets larger than 1500 bytes as if they were 1500 bytes in length.

`Path` messages for the ***Sherry-to-Char*** LSP in Figure 7.2 are received by the Merlot router. We can view the contents of these messages in the output of a `traceoptions` file:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

Within the object, we see that no bandwidth requests are configured for this LSP. This is evident by the `rate 0bps` and `size 0bps` output on the router. The infinite peak data rate associated with the LSP is represented as `peak Infbps`, while the minimum (`m`) and maximum (`M`) packet sizes are also displayed.

The Label Object

The *Label object* is contained in `Resv` messages and advertises an MPLS label value upstream to the neighboring router. The label is used by that upstream router to forward packets within the LSP to the advertising router. The allocation of a label value for inclusion in this object must be prompted by a request for label in the `Path` message for this LSP. Figure 7.12 displays for fields for the Label object:

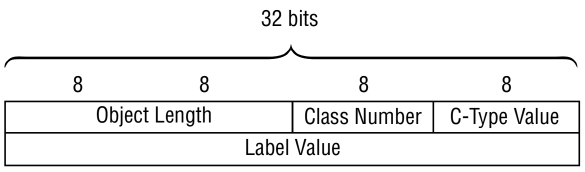
Object Length (2 octets) This field displays the total length of the RSVP object. For the Label object, a constant value of 8 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Label object falls within the Label class, which uses a value of 16.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Label object uses a value of 1 for its C-Type.

Label Value (4 octets) This field contains the label value advertised by the downstream router. The value is right-justified within the 32-bit field and occupies bits 0 through 19.

FIGURE 7.12 The Label object



We can see the Label object in a `Resv` message received by the Merlot router in Figure 7.2:

```
Apr 24 10:56:24 RSVP recv Resv 10.222.45.2->10.222.45.1 Len=120 so-0/1/2.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.45.2/0x08550264
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  3
Apr 24 10:56:24   RecRoute Len 12  10.222.45.2
```

The egress router of Chardonnay (10.222.45.2) advertises a label value of 3 to Merlot, which requests that PHP be performed for this LSP. Merlot stores this information in its RSVP soft state, allocates its own label value, and advertises a new `Resv` message upstream to Chianti:

```
Apr 24 10:56:24 RSVP send Resv 10.222.1.2->10.222.1.1 Len=128 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.2/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   Style    Len  8 FF
Apr 24 10:56:24   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Label    Len  8  100001
Apr 24 10:56:24   RecRoute Len 20  10.222.1.2 10.222.45.2
```

The Label Request Object

The *Label Request object* is contained in `Path` messages and prompts the advertisement of label values upstream in `Resv` messages. You can use three different forms of Label Request objects in an MPLS network. The first request type doesn't specify a specific range of labels, which allows the downstream routers to allocate any possible value. This request type is the only form used by the JUNOS software. The other two label request types allow for an allocation from either an ATM or Frame Relay label range. Figure 7.13 shows the format of the Label Request object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object. For the Label Request object, a constant value of 8 is placed in this field.

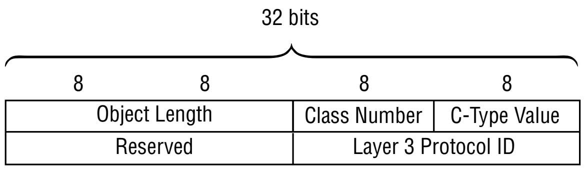
Class Number (1 octet) The value that represents the object's class is placed into this field. The Label Request object falls within the Label Request class, which uses a value of 19.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Label Request object uses a value of 1 for its C-Type.

Reserved (2 octets) This field is not used and is set to a constant value of 0x0000.

Layer 3 Protocol ID (2 octets) This field contains information about the Layer 3 protocol carried in the LSP. The standard EtherType value of 0x0800 is used in this field to represent IPv4.

FIGURE 7.13 The Label Request object



Referring back to the LSP created in Figure 7.2, we can view the Label Request object in the `Path` message received by Merlot:

```
Apr 24 10:56:24 RSVP recv Path 192.168.16.1->192.168.32.1 Len=228 so-0/1/0.0
Apr 24 10:56:24   Session7 Len 16 192.168.32.1(port/tunnel ID 44214) Proto 0
Apr 24 10:56:24   Hop      Len 12 10.222.1.1/0x084ec198
Apr 24 10:56:24   Time     Len  8 30000 ms
Apr 24 10:56:24   SessionAttribute Len 28 Prio (7,0) flag 0x0 "Sherry-to-Char"
Apr 24 10:56:24   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
Apr 24 10:56:24   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
Apr 24 10:56:24   ADspec   Len 48
Apr 24 10:56:24   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
Apr 24 10:56:24   LabelRequest Len  8 EtherType 0x800
Apr 24 10:56:24   Properties Len 12 Primary path
Apr 24 10:56:24   RecRoute Len 20  10.222.1.1 10.222.29.1
```

The upstream router of Chianti (10.222.1.1) has requested that a label be allocated for the ***Sherry-to-Char*** LSP being established by this message. The `EtherType 0x800` notation informs the Merlot router that IPv4 data traffic will be using this LSP once it is operational.

The Explicit Route Object

The *Explicit Route object* (ERO) is contained in `Path` messages and allows the ingress router to specify the path of the LSP through the network by containing a list of nodes, defined as a subobject, through which the RSVP messages must pass. These nodes are in the form of an IPv4 prefix, an IPv6 prefix, or an Autonomous System number, with the JUNOS software supporting just IPv4 prefixes. When the `Path` message is received by a router, the ERO is examined to first determine if the first node is strictly or loosely defined. The local router then determines whether the first node address in the ERO equals the local router's address. At this point, one of two possible scenarios exists. If the first node in the ERO is loosely defined and the local address is not equal to the address listed in the ERO, the local router processes the `Path` message and forwards it to the address listed in the ERO.

The second scenario occurs when the first ERO address is strictly defined, in which case the local address must match the ERO address in order for the local router to process the `Path` message. If the address values do not match, the local router does not process the message and instead generates a `PathErr` message and forwards the error message back to the ingress router. Assuming that the strict hop is correctly defined, the local router creates its local Path State Block and prepares to forward the message further downstream. Before doing so, however, the first address in the ERO is removed to allow the next downstream router to perform the same sanity checks we've discussed here.

Figure 7.14 displays the fields of the Explicit Route object when it contains IPv4 prefixes. The various fields are as follows:

Object Length (2 octets) This field displays the total length of the RSVP object.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Explicit Route Object falls within the Explicit Route class, which uses a value of 20.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Explicit Route object uses a value of 1 for its C-Type.

L bit and Type (1 octet) The most significant bit in this 1-octet field is called the L bit and is used to denote if the included address is loosely or strictly defined. When the bit is set to a value of 1, the address is a loose node. Strict node addresses are defined by a value of 0 for this bit.

The remaining 7 bits in the field encode the type of address contained in the subobject. For an IPv4 address, a constant value of 0x01 is used in this field.

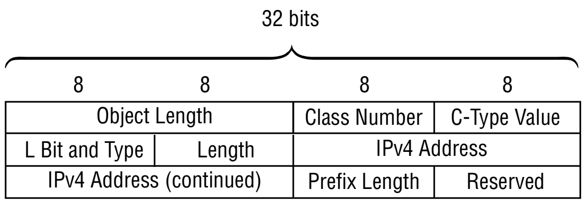
Length (1 octet) This field displays the length of the subobject, including the type and length fields. For an IPv4 address, a constant value of 0x08 is used in this field.

IPv4 Address (4 octets) This field contains the IPv4 address listed as a node along the path.

Prefix Length (1 octet) This field displays the prefix length of the preceding IPv4 address. For a host address, the default configuration, a constant value of 32 is placed in this field.

Reserved (1 octet) This field is not used and is set to a constant value of 0x00.

FIGURE 7.14 The Explicit Route object



The Sherry router in Figure 7.2 generates a `Path` message for the ***Sherry-to-Char*** LSP, which contains the strict-hop address in the ERO. We can view this information in the output of a `traceoptions` file on Sherry:

```
May 12 12:09:17 RSVP send Path 192.168.16.1->192.168.32.1 Len=224 ge-0/2/0.0
May 12 12:09:17   Session7 Len 16 192.168.32.1(port/tunnel ID 24025) Proto 0
May 12 12:09:17   Hop      Len 12 10.222.29.1/0x08528264
May 12 12:09:17   Time     Len  8 30000 ms
May 12 12:09:17   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 12:09:17   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 12:09:17   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 12:09:17   ADspec   Len 48
May 12 12:09:17   SrcRoute Len 28  10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
May 12 12:09:17   LabelRequest Len  8 EtherType 0x800
May 12 12:09:17   Properties Len 12 Primary path
May 12 12:09:17   RecRoute Len 12  10.222.29.1
```

The `SrcRoute` output contains the ERO information within the `Path` message. Each of the three downstream routers is specified as a strict-hop node along the path. The LSP should first traverse the router whose address is 10.222.29.2 (Chianti); it will then be sent to 10.222.1.2 (Merlot) before reaching the final address of 10.222.45.2 (Chardonnay). Because the final address in the ERO equals the address of the egress router, 192.168.36.1, Chardonnay stops processing the `Path` message and generates a corresponding `Resv` message for transmission upstream.

The Record Route Object

The *Record Route object* (RRO) may be contained in either a `Path` or a `Resv` message. The RRO contains subobjects that describe the nodes through which the message has passed. These node addresses are in the form of an IPv4 prefix or an IPv6 prefix. In addition, a label value for the RSVP session may be recorded in a subobject within the RRO. Currently, the JUNOS software supports only an IPv4 prefix address in the Record Route object.

The object is useful for troubleshooting the operation of an LSP, but it mainly allows routers in the network to detect loops during the setup of an LSP. When an RSVP router receives a message that contains a Record Route object, the contents of that object are examined. If the local router finds one of its local addresses in the object fields, a loop has formed and the message is not processed any further. In addition, the local router generates an error message (either a `PathErr` or a `ResvErr`) and sends it to the router from which it received the original message. The fields used for IPv4 addresses in the Record Route object are displayed in Figure 7.15 and include the following:

Object Length (2 octets) This field displays the total length of the RSVP object.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Record Route object falls within the Record Route class, which uses a value of 21.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Record Route object uses a value of 1 for its C-Type.

Type (1 octet) This field encodes the type of address contained in the sub-object. For an IPv4 address, a constant value of 0x01 is used in this field.

Length (1 octet) This field displays the length of the subobject, including the type and length fields. For an IPv4 address, a constant value of 0x08 is used in this field.

IPv4 Address (4 octets) This field contains a 32-bit host address that represents the router processing the RSVP message. While any valid network-accessible address is allowed here, the JUNOS software places the address of the router's outgoing interface in this field.

Prefix Length (1 octet) This field displays the prefix length of the preceding IPv4 address. A constant value of 32 is placed in this field.

Flags (1 octet) This field contains flags that alert other routers in the network about the capabilities or conditions of the local node. Currently, four flag values are defined:

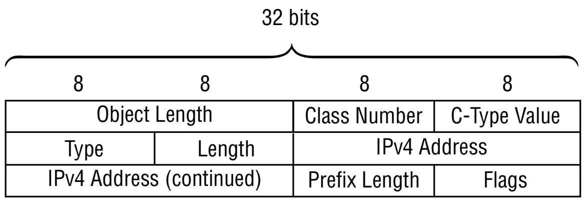
Bit 0 This flag bit, 0x01, is set to indicate that the next downstream link from the router is protected by a local repair mechanism, such as fast reroute link protection. The flag is set only when the Session Attribute object requests that link protection be enabled for the LSP.

Bit 1 This flag bit, 0x02, is set to indicate that the local router is actively using a repair mechanism to maintain the LSP due to some outage condition.

Bit 2 This flag bit, 0x04, is set to indicate that the local router has a backup path enabled that provides the same bandwidth reservation guarantees as established for an LSP protected through a local protection scheme.

Bit 3 This flag bit, 0x08, is set to indicate that the next downstream node as well as the downstream link is protected by a local repair mechanism, such as fast reroute node protection. This flag is set only when the next downstream node is protected against failure and the Session Attribute object requests that node protection be enabled for the LSP.

FIGURE 7.15 The Record Route object



The Merlot router is receiving both `Path` and `Resv` messages for the ***Sherry-to-Char*** LSP in Figure 7.2. A Record Route object is present in both types of messages:

```
May 12 14:04:12 RSVP recv Path 192.168.16.1->192.168.32.1 Len=224 so-0/1/0.0
May 12 14:04:12   Session7 Len 16 192.168.32.1(port/tunnel ID 24025) Proto 0
May 12 14:04:12   Hop      Len 12 10.222.1.1/0x085280cc
May 12 14:04:12   Time     Len  8 30000 ms
May 12 14:04:12   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 14:04:12   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 14:04:12   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 14:04:12   ADspec   Len 48
May 12 14:04:12   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
May 12 14:04:12   LabelRequest Len  8 EtherType 0x800
May 12 14:04:12   Properties Len 12 Primary path
May 12 14:04:12   RecRoute Len 20  10.222.1.1 10.222.29.1

May 12 14:04:12 RSVP recv Resv 10.222.45.2->10.222.45.1 Len=120 so-0/1/2.0
May 12 14:04:12   Session7 Len 16 192.168.32.1(port/tunnel ID 24025) Proto 0
May 12 14:04:12   Hop      Len 12 10.222.45.2/0x08572264
May 12 14:04:12   Time     Len  8 30000 ms
May 12 14:04:12   Style    Len  8 FF
May 12 14:04:12   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 14:04:12   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 14:04:12   Label    Len  8  3
May 12 14:04:12   RecRoute Len 12  10.222.45.2
```

Within each message type, the `RecRoute` output contains the information encoded within the RRO. When we examine the `Path` message, which is first in the output, we find that the 10.222.29.1 router (Sherry) processed the message first followed by the 10.222.1.1 router (Chianti). Because none of Merlot's local interfaces appear in the RRO, the `Path` message is processed locally and Merlot adds its own downstream interface address (10.22.45.1) to the RRO before sending the message to Chardonnay.

When Merlot receives a Resv message for the LSP from Chardonnay, the RRO contains just the address of the egress router, 10.222.45.2. As with the Path message, Merlot examines the object in the Resv message to ensure that one of its local addresses doesn't appear. After performing this check, Merlot processes the message and sends a corresponding Resv message upstream to Chianti after adding its outgoing interface address of 10.222.1.2 to the object.

The Detour Object

The *Detour object* is contained in a Path message, which establishes a fast reroute detour in the network. This Path message inherits the Session, Sender-Template, and Session Attribute objects from the established LSP. This allows the detour path to be associated with the main RSVP session on all possible routers. The Detour object itself lists the originating node of the Path message, the point of local repair, and the address of the downstream node being protected against failures. These ID values may be repeated in a single Detour object when an RSVP router merges multiple detour paths as they head toward the egress router.

We discuss the functionality of an LSP supporting fast reroute in greater detail in Chapter 8, "Advanced MPLS."

Figure 7.16 displays the fields of the Detour object, which include the following:

Object Length (**2 octets**) This field displays the total length of the RSVP object.

Class Number (**1 octet**) The value that represents the object's class is placed into this field. The Detour object falls within the Detour class, which uses a value of 63.

C-Type Value (**1 octet**) The specific type of object within its overall class is displayed in this field. The Detour object uses a value of 7 for its C-Type.

Local Repair Node ID (**4 octets**) This field encodes the address of the router creating the detour path through the network. While any possible address on the local router may be used, the JUNOS software places the address of the outgoing interface in this field.

Avoid Node ID (**4 octets**) This field encodes the router ID of the downstream node that is being protected against failures.

FIGURE 7.16 The Detour object

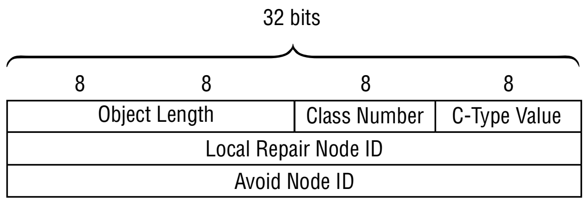


FIGURE 7.17 Fast reroute LSP establishment

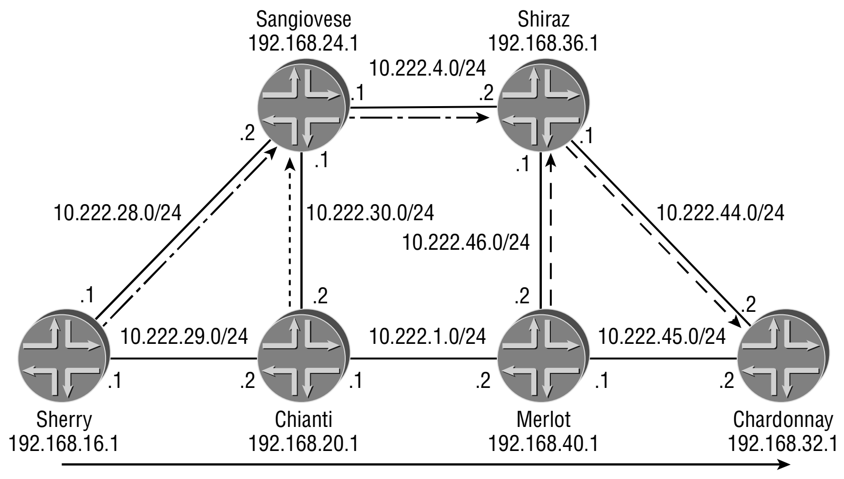


Figure 7.17 shows the ***Sherry-to-Char*** LSP now configured to support fast reroute. Once the LSP becomes established through the Chianti, Merlot, and Chardonnay routers, the ingress router and all transit routers create detour paths through the network to protect against link and node failures. Sherry, the ingress router, creates a detour through the Sangiovese and Shiraz routers before reaching the egress router of Chardonnay. In the output of a `traceoptions` file on Sherry, we first see the `Path` message establishing the LSP followed by the generation of a second `Path` message establishing the detour path:

```
May 12 17:41:01 RSVP send Path 192.168.16.1->192.168.32.1 Len=244 ge-0/2/0.0
May 12 17:41:01   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:41:01   Hop      Len 12 10.222.29.1/0x08528264
May 12 17:41:01   Time     Len  8 30000 ms
May 12 17:41:01   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:41:01   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:41:01   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:41:01   ADspec   Len 48
May 12 17:41:01   SrcRoute Len 28  10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
May 12 17:41:01   LabelRequest Len  8 EtherType 0x800
May 12 17:41:01   Properties Len 12 Primary path
May 12 17:41:01   RecRoute Len 12  10.222.29.1
May 12 17:41:01   FastReroute Len 20 Prio(7,0) Hop 6 BW 0bps
May 12 17:41:01     Include 0x00000000 Exclude 0x00000000

May 12 17:41:04 RSVP send Path 192.168.16.1->192.168.32.1 Len=236 at-0/1/0.0
May 12 17:41:04   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:41:04   Hop      Len 12 10.222.28.1/0x085280cc
May 12 17:41:04   Time     Len  8 30000 ms
May 12 17:41:04   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:41:04   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:41:04   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:41:04   ADspec   Len 48
May 12 17:41:04   SrcRoute Len 28  10.222.28.2 S 10.222.4.2 S 10.222.44.2 S
May 12 17:41:04   LabelRequest Len  8 EtherType 0x800
May 12 17:41:04   Properties Len 12 Primary path
May 12 17:41:04   RecRoute Len 12  10.222.28.1
May 12 17:41:04   Detour   Len 12  Branch from 10.222.28.1 to avoid 192.168.20.1
```

Within the detour `Path` message type, we see the same information in the `Session7`, `SessionAttribute`, and `Sender7` objects. The configured ERO (`SrcRoute`) details the path through the network for the detour, whereas the Detour object itself lists the outgoing interface on the Sherry router (10.222.28.1) and the protected downstream node of Chianti (192.168.20.1).

Chianti, the first transit router, also creates a detour path to the egress router to protect the LSP against failures associated with the Merlot router. This detour also uses Sangiovese, Shiraz, and Chardonnay as its network path. When the Sangiovese router receives the detour `Path` messages from both Sherry and Chianti, it finds that they belong to the same RSVP session. This allows Sangiovese to merge the detour paths together and send a single `Path` message downstream to Shiraz. During this merge process, the Detour object lists both Sherry (10.222.28.1) and Chianti (10.222.30.2) as local repair nodes. The output of a `traceoptions` file on Sangiovese shows the receipt of these two `Path` messages and the generation of the merged `Path` message sent downstream:

```
May 12 17:43:55 RSVP recv Path 192.168.16.1->192.168.32.1 Len=236 at-0/1/0.0
May 12 17:43:55   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:43:55   Hop      Len 12 10.222.28.1/0x085280cc
May 12 17:43:55   Time     Len  8 30000 ms
May 12 17:43:55   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:43:55   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:43:55   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:43:55   ADspec   Len 48
May 12 17:43:55   SrcRoute Len 28  10.222.28.2 S 10.222.4.2 S 10.222.44.2 S
May 12 17:43:55   LabelRequest Len  8 EtherType 0x800
May 12 17:43:55   Properties Len 12 Primary path
May 12 17:43:55   RecRoute Len 12  10.222.28.1
May 12 17:43:55   Detour   Len 12  Branch from 10.222.28.1 to avoid 192.168.20.1

May 12 17:43:55 RSVP recv Path 192.168.16.1->192.168.32.1 Len=244 so-0/2/0.0
May 12 17:43:55   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:43:55   Hop      Len 12 10.222.30.2/0x08528330
May 12 17:43:55   Time     Len  8 30000 ms
May 12 17:43:55   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:43:55   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:43:55   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:43:55   ADspec   Len 48
May 12 17:43:55   SrcRoute Len 28  10.222.30.1 S 10.222.4.2 S 10.222.44.2 S
May 12 17:43:55   LabelRequest Len  8 EtherType 0x800
May 12 17:43:55   Properties Len 12 Primary path
May 12 17:43:55   RecRoute Len 20  10.222.30.2 10.222.29.1
May 12 17:43:55   Detour   Len 12  Branch from 10.222.30.2 to avoid 192.168.40.1

May 12 17:43:55 RSVP send Path 192.168.16.1->192.168.32.1 Len=244 fe-0/0/2.0
May 12 17:43:55   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:43:55   Hop      Len 12 10.222.4.1/0x085650cc
May 12 17:43:55   Time     Len  8 30000 ms
May 12 17:43:55   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:43:55   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:43:55   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:43:55   ADspec   Len 48
May 12 17:43:55   SrcRoute Len 20  10.222.4.2 S 10.222.44.2 S
May 12 17:43:55   LabelRequest Len  8 EtherType 0x800
May 12 17:43:55   Properties Len 12 Primary path
May 12 17:43:55   RecRoute Len 20  10.222.4.1 10.222.28.1
May 12 17:43:55   Detour   Len 20  Branch from 10.222.28.1 to avoid 192.168.20.1
May 12 17:43:55     Branch from 10.222.30.2 to avoid 192.168.40.1
```

Finally, the Shiraz router also performs a merge of detour paths in our sample network. The first detour arrives from the Sangiovese router, and the second arrives from Merlot. We see the merged Detour object in the `Path` message sent by Shiraz to Chardonnay:

```
May 12 17:41:01 RSVP send Path 192.168.16.1->192.168.32.1 Len=260 so-0/1/0.0
May 12 17:41:01   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:41:01   Hop      Len 12 10.222.44.1/0x085a7198
May 12 17:41:01   Time     Len  8 30000 ms
May 12 17:41:01   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:41:01   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:41:01   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:41:01   ADspec   Len 48
May 12 17:41:01   SrcRoute Len 12  10.222.44.2 S
May 12 17:41:01   LabelRequest Len  8 EtherType 0x800
May 12 17:41:01   Properties Len 12 Primary path
May 12 17:41:01   RecRoute Len 36 10.222.44.1 10.222.46.2 10.222.1.1 10.222.29.1
May 12 17:41:01   Detour   Len 28  Branch from 10.222.46.2 to avoid 192.168.32.1
May 12 17:41:01     Branch from 10.222.28.1 to avoid 192.168.20.1
May 12 17:41:01     Branch from 10.222.30.2 to avoid 192.168.40.1
```

The Fast Reroute Object

The *Fast Reroute object* is contained in `Path` messages sent along the path of an established LSP. It alerts all downstream routers that the ingress router desires protection along the LSP's path. Each router along the LSP, with the exception of the egress, then creates a detour path around the next downstream node using the Detour object. Information within the Fast Reroute object allows each of the routers to consult a local traffic engineering database for calculating a path to the egress router. This information includes a bandwidth reservation, a hop count, LSP priority values, and administrative group knowledge. We discuss the mechanics of fast reroute in greater depth in Chapter 8.

The format of the Fast Reroute object is displayed in Figure 7.18. The field definitions include the following:

Object Length (2 octets) This field displays the total length of the RSVP object. A constant value of 20 is placed in this field.

Class Number (1 octet) The value that represents the object's class is placed into this field. The Fast Reroute object falls within the Fast Reroute class, which uses a value of 205.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The Fast Reroute object uses a value of 1 for its C-Type.

Setup Priority (1 octet) This field contains the priority of the LSP used for assigning resources during its establishment in the network. Possible values range from 0 through 7, with 0 representing the strongest priority value and 7 the weakest priority value. The JUNOS software uses a setup priority value of 7 by default.

Hold Priority (1 octet) This field contains the priority of the LSP used for maintaining resources after becoming established in the network. Possible values range from 0 through 7, with 0 representing the strongest priority value and 7 the weakest priority value. The JUNOS software uses a hold priority value of 0 by default.

Hop Limit (1 octet) This field displays the total number of transit hops a detour path may take through the network, excluding the local repair node and any router performing a detour merge operation. For example, a hop limit value of 2 means that the detour can leave the local repair node and transit two other routers before being merged or reaching the egress router.

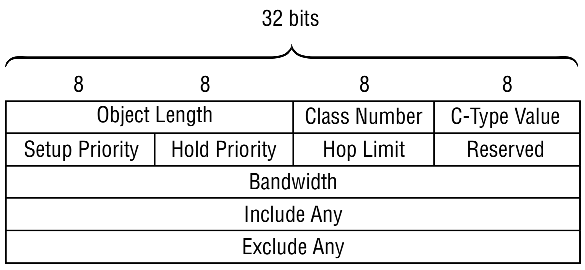
Reserved (1 octet) This field is not used and is set to a constant value of 0x00.

Bandwidth (4 octets) When populated, this field displays the bandwidth reservation (in bytes per second) that should be performed for all detour paths. By default, the JUNOS software places a value of 0 in this field. You may alter this default by configuring a bandwidth within the `fast-reroute` definition of the LSP.

Include Any (4 octets) This field contains information pertaining to network links that are assigned a particular administrative group. When a group value is placed in this field, each network link along the detour path must be assigned to that group. A value of 0 in this field means that no group values are required and that all network links may be used.

Exclude Any (4 octets) This field contains information pertaining to network links that are assigned a particular administrative group. When a group value is placed in this field, each network link along the detour path must *not* be assigned to that group. A value of 0 in this field means that any network link may be used.

FIGURE 7.18 The Fast Reroute object



Using the sample network shown in Figure 7.17 as a guide, we examine the `Path` message generated by the Sherry router for the ***Sherry-to-Char*** LSP:

```
May 12 17:41:01 RSVP send Path 192.168.16.1->192.168.32.1 Len=244 ge-0/2/0.0
May 12 17:41:01   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:41:01   Hop      Len 12 10.222.29.1/0x08528264
May 12 17:41:01   Time     Len  8 30000 ms
May 12 17:41:01   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:41:01   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:41:01   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:41:01   ADspec   Len 48
May 12 17:41:01   SrcRoute Len 28  10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
May 12 17:41:01   LabelRequest Len  8 EtherType 0x800
May 12 17:41:01   Properties Len 12 Primary path
May 12 17:41:01   RecRoute Len 12  10.222.29.1
May 12 17:41:01   FastReroute Len 20 Prio(7,0) Hop 6 BW 0bps
May 12 17:41:01     Include 0x00000000 Exclude 0x00000000
```

The `FastReroute` output displays the information contained in the Fast Reroute object. The default setup (7) and hold (0) priority values are requested, and the hop limit is set to 6 hops. No bandwidth reservation is assigned to the detour paths, and all possible links in the network are usable by the detours. This is revealed by examining the `Include` and `Exclude` fields and finding all zero values.

The LSP-Tunnel Session Attribute Object

The *LSP-Tunnel Session Attribute object* is contained in a `Path` message and is used by the ingress router to advertise the priority values associated with the LSP. In addition, the name of the RSVP session is included in the object, which allows for easier troubleshooting on transit and egress routers. Finally, information concerning the operation of the LSP and its resource reservations is contained here as well. Figure 7.19 displays the format of the Session Attribute object, which includes the following fields:

Object Length (2 octets) This field displays the total length of the RSVP object.

Class Number (1 octet) The value that represents the object's class is placed into this field. The LSP-Tunnel Session Attribute object falls within the Session Attribute class, which uses a value of 207.

C-Type Value (1 octet) The specific type of object within its overall class is displayed in this field. The LSP-Tunnel Session Attribute object uses a value of 7 for its C-Type.

Setup Priority (1 octet) This field contains the priority of the LSP used for assigning resources during its establishment in the network. Possible values range from 0 through 7, with 0 representing the strongest priority value and 7 the weakest priority value. The JUNOS software uses a setup priority value of 7 by default.

Hold Priority (1 octet) This field contains the priority of the LSP used for maintaining resources after becoming established in the network. Possible values range from 0 through 7, with 0 representing the strongest priority value and 7 the weakest priority value. The JUNOS software uses a hold priority value of 0 by default.

Flags (1 octet) This field contains flags that alert other routers in the network about the capabilities of the LSP and its resource reservations. Currently, five flag values are defined:

Bit 0 This flag bit, 0x01, is set to permit downstream routers to use a local repair mechanism, such as fast reroute link protection, allowing transit routers to alter the explicit route of the LSP.

Bit 1 This flag bit, 0x02, is set to request that label recording be performed along the LSP path. This means that each downstream node should place its assigned label into the Record Route object in the `Resv` message.

Bit 2 This flag bit, 0x04, is set to indicate that the egress router should use the Shared Explicit reservation style for the LSP. This allows the ingress router to reroute the primary path of the LSP without first releasing the LSP's resources.

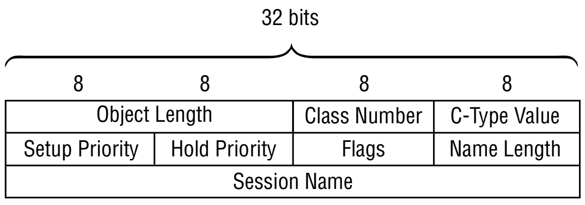
Bit 3 This flag bit, 0x08, is set to indicate that each router along the LSP should reserve bandwidth for its fast reroute detour path. The detour paths in the network use this bit value only when the Fast Reroute object is omitted from the `Path` message created by the ingress router.

Bit 4 This flag bit, 0x10, is set to permit downstream routers to use a node repair mechanism, such as fast reroute node protection. Each router should then calculate a detour path that protects the LSP from a failure of the next downstream node.

Name Length (1 octet) This field displays the length of the Session Name field that follows.

Session Name (Variable) This variable-length field contains the configured ASCII name of the LSP. Its inclusion in a `Path` message allows each router along the path to display the LSP name in relevant `show` commands.

FIGURE 7.19 The LSP-Tunnel Session Attribute object



The Merlot router is receiving `Path` messages from Chianti for the ***Sherry-to-Char*** LSP in Figure 7.17. The Session Attribute object appears within the message as so:

```
May 12 17:40:51 RSVP recv Path 192.168.16.1->192.168.32.1 Len=244 so-0/1/0.0
May 12 17:40:51   Session7 Len 16 192.168.32.1(port/tunnel ID 24027) Proto 0
May 12 17:40:51   Hop      Len 12 10.222.1.1/0x085280cc
May 12 17:40:51   Time     Len  8 30000 ms
May 12 17:40:51   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 12 17:40:51   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 12 17:40:51   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 12 17:40:51   ADspec   Len 48
May 12 17:40:51   SrcRoute Len 20  10.222.1.2 S 10.222.45.2 S
May 12 17:40:51   LabelRequest Len  8 EtherType 0x800
May 12 17:40:51   Properties Len 12 Primary path
May 12 17:40:51   RecRoute Len 20  10.222.1.1 10.222.29.1
May 12 17:40:51   FastReroute Len 20 Prio(7,0) Hop 6 BW 0bps
May 12 17:40:51     Include 0x00000000 Exclude 0x00000000
```

The information within the object informs us that the default JUNOS software setup (7) and hold (0) priority values are used for this LSP and that no flags have been set. In addition, the inclusion of the LSP name allows Merlot to display the string in the output of the `show mpls lsp` command:

```
user@Merlot> 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: 1 sessions
To              From            State Rt Style Labelin Labelout LSPname
192.168.32.1    192.168.16.1    Up     1  1 FF  100112        3 Sherry-to-Char
Total 1 displayed, Up 1, Down 0
```

RSVP Sessions

When an LSP is established in the network, it is assigned certain resources, including a label value and a bandwidth reservation when it's requested. In addition, the use of fast reroute for the LSP means that each detour path created in the network needs to be associated with the protected LSP. All of this information is connected by a common set of data referred to as an *RSVP session*. Specifically, a session is identified by the unique combination of information in `Path` and `Resv` message objects. Within a `Path` message, the Session, Sender-Template, and Session Attribute objects define a session, while the Session and Filter-Spec objects are used in a `Resv` message.

The Sherry router is currently the ingress router for an LSP in an MPLS network. It generates the following `Path` message and forwards it to the first transit router in the path:

```
May 13 13:20:30 RSVP send Path 192.168.16.1->192.168.32.1 Len=244 ge-0/2/0.0
May 13 13:20:30   Session7 Len 16 192.168.32.1(port/tunnel ID 24039) Proto 0
May 13 13:20:30   Hop      Len 12 10.222.29.1/0x08528264
May 13 13:20:30   Time     Len  8 30000 ms
May 13 13:20:30   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 13 13:20:30   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 13 13:20:30   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 13 13:20:30   ADspec   Len 48
May 13 13:20:30   SrcRoute Len 28  10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
May 13 13:20:30   LabelRequest Len  8 EtherType 0x800
May 13 13:20:30   Properties Len 12 Primary path
May 13 13:20:30   RecRoute Len 12  10.222.29.1
May 13 13:20:30   FastReroute Len 20 Prio(7,0) Hop 6 BW 0bps
May 13 13:20:30     Include 0x00000000 Exclude 0x00000000
```

The `Session7` object defines the egress router address as 192.168.32.1 and assigns a tunnel ID value of 24039 to the LSP. The ingress router information is displayed in the `Sender7` object, where we see the 192.168.16.1 address and an LSP ID value of 1. The LSP itself is assigned the name ***`Sherry-to-Char`***, which is contained in the `SessionAttribute` object. This same information is contained in the `Resv` message Sherry receives from the next downstream node:

```
May 13 13:20:30 RSVP recv Resv 10.222.29.2->10.222.29.1 Len=136 ge-0/2/0.0
May 13 13:20:30   Session7 Len 16 192.168.32.1(port/tunnel ID 24039) Proto 0
May 13 13:20:30   Hop      Len 12 10.222.29.2/0x08528264
May 13 13:20:30   Time     Len  8 30000 ms
May 13 13:20:30   Style    Len  8 FF
May 13 13:20:30   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 13 13:20:30   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
May 13 13:20:30   Label    Len  8  100144
May 13 13:20:30   RecRoute Len 28  10.222.29.2(flag=0x9)
May 13 13:20:30     10.222.1.2(flag=0x1) 10.222.45.2
```

The egress router address and tunnel ID value are once again contained in the `Session7` object. This is the same object data used in the `Path` message. Information concerning the ingress router is now placed into the `Filter7` object. Here we see the LSP ID value of 1 associated with the router address of 192.168.16.1.

The receipt of the `Resv` message by Sherry completes the establishment of the LSP. We can view the RSVP session details in the output of the `show rsvp session detail ingress` command as so:

```
user@Sherry> show rsvp session detail ingress
Ingress RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100144
  Resv style: 1 FF, Label in: -, Label out: 100144
  Time left:    -,  Since: Tue May 13 13:20:18 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 24039 protocol 0
  FastReroute desired
  PATH rcvfrom: localclient
  PATH sentto: 10.222.29.2 (ge-0/2/0.0) 18 pkts
  RESV rcvfrom: 10.222.29.2  (ge-0/2/0.0) 20 pkts
  Explct route: 10.222.29.2 10.222.1.2 10.222.45.2
  Record route: <self>  10.222.29.2  10.222.1.2  10.222.45.2
    Detour is Up
    Detour PATH sentto: 10.222.28.2 (at-0/1/0.0) 17 pkts
    Detour RESV rcvfrom: 10.222.28.2  (at-0/1/0.0) 17 pkts
    Detour Explct route: 10.222.28.2 10.222.4.2 10.222.44.2
    Detour Record route: <self>  10.222.28.2  10.222.4.2  10.222.44.2
    Detour Label out: 100192
Total 1 displayed, Up 1, Down 0
```

The addresses of the ingress and egress routers are clearly visible in the router output, as is the name of the LSP. The command also displays, however, the ID numbers assigned by Sherry for this session within the `Port number` section. The tunnel ID value of 24039 appears as the `receiver`, while the LSP ID value 1 appears as the `sender`. Finally, the ingress router has requested that the LSP be protected through a fast reroute mechanism, as shown by the `FastReroute` desired output. This means that Sherry generates a `Path` message for the creation of the detour path. When the detour is established, Sherry receives a `Resv` message from the first downstream node along the detour. Both of these messages are associated with the original RSVP session through the inclusion of the appropriate objects:

```
May 13 13:20:30 RSVP send Path 192.168.16.1->192.168.32.1 Len=236 at-0/1/0.0
May 13 13:20:30   Session7 Len 16 192.168.32.1(port/tunnel ID 24039) Proto 0
May 13 13:20:30   Hop      Len 12 10.222.28.1/0x085280cc
May 13 13:20:30   Time     Len  8 30000 ms
May 13 13:20:30   SessionAttribute Len 24 Prio (7,0) flag 0x0 "Sherry-to-Char"
May 13 13:20:30   Sender7  Len 12 192.168.16.1(port/lsp ID  1)
May 13 13:20:30   Tspec    Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 13 13:20:30   ADspec   Len 48
May 13 13:20:30   SrcRoute Len 28  10.222.28.2 S 10.222.4.2 S 10.222.44.2 S
May 13 13:20:30   LabelRequest Len  8 EtherType 0x800
May 13 13:20:30   Properties Len 12 Primary path
May 13 13:20:30   RecRoute Len 12  10.222.28.1
May 13 13:20:30   Detour   Len 12  Branch from 10.222.28.1 to avoid 192.168.20.1

May 13 13:20:39 RSVP recv Resv 10.222.28.2->10.222.28.1 Len=136 at-0/1/0.0
May 13 13:20:39   Session7 Len 16 192.168.32.1(port/tunnel ID 24039) Proto 0
May 13 13:20:39   Hop      Len 12 10.222.28.2/0x085280cc
May 13 13:20:39   Time     Len  8 30000 ms
May 13 13:20:39   Style    Len  8 FF
May 13 13:20:39   Flow     Len 36 rate 0bps size 0bps peak Infbps m 20 M 1500
May 13 13:20:39   Filter7  Len 12 192.168.16.1(port/lsp ID  1)
May 13 13:20:39   Label    Len  8  100192
May 13 13:20:39   RecRoute Len 28  10.222.28.2 10.222.4.2 10.222.44.2
```

The Label Distribution Protocol

The second dynamic method of establishing an LSP within the JUNOS software is the *Label Distribution Protocol* (LDP). Unlike the extended version of RSVP, no traffic engineering capabilities are available with LDP, and each label-switched path follows the Interior Gateway Protocol (IGP) shortest path through the network. Each LDP-speaking router advertises an address reachable via an MPLS label into the LDP domain. This label information is exchanged by neighbors in a hop-by-hop fashion so that every router in the network becomes an ingress router to every other router in the network. The end result of this process is a full mesh of LSPs in the domain. Before this can occur, however, each set of adjacent routers forms an LDP neighbor relationship.

Becoming LDP Neighbors

After the protocol is enabled on a local router, it begins sending *LDP Hello messages* on all of its operational interfaces. These messages are addressed to the 224.0.0.2 /32 well-known destination address and are sent using UDP port 646. The Hello message, as well as all other LDP messages, is encoded using a type, length, value (TLV) paradigm. Each message contains a common LDP header, which is followed by a set of fields describing the message itself. The body of the message contains multiple TLVs, some of which are mandatory, whereas others are optional. Figure 7.20 displays the format of the LDP Hello message, which includes a single mandatory TLV, the hello parameters TLV. The JUNOS software uses two optional TLVs in the Hello message to describe the local router and its properties. The message's fields are as follows:

LDP Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to a constant value of 1.

PDU Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the PDU.

LDP ID (6 octets) This field contains a unique LDP ID, which describes the label space used by the local router. The first 4 octets of the ID value represent the advertising node as a unique device in the network. The router ID of the node is placed in this portion of the ID. The final 2 octets of the ID are used to describe the method of label allocation used by the local router. When a per-router allocation system is in use, as is the JUNOS software default, these octets are set to a value of 0. The LDP ID is displayed in a format of 192.168.1.1:0.

Message Type (2 octets) The first bit in this field is defined as the U, or unknown, bit. It is designed as a method for telling routers how to handle a message that is unknown to the local router. When the bit is clear (set to 0), the router must return an error message to the sender if the message type is not recognized. If the bit is set to a value of 1, the local router may ignore the unknown TLV message silently while continuing to process the remainder of the message. For an LDP Hello message, the U bit is set to a value of 0.

The remaining bits in this field represent the type of message contained in the packet. Hello messages place a constant value of 0x0100 in this field.

Message Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the message.

Message ID (4 octets) This field contains a 32-bit value generated by the advertising router, which uniquely identifies this particular message.

Hello TLV Type (2 octets) The first bit in this field is also defined as a U bit. As before, it defines how a receiving router handles an unknown TLV. The hello parameters TLV uses a value of 0 for this bit, requiring all routers to return an error message to the source. The second bit in the field is the F, or forward, bit. It is used only when the U bit is set to a value of 1 and the LDP message is forwarded beyond the receiving router. When the F bit is cleared, the receiving router doesn't forward the unknown TLV along with the message. When the F bit is set to a value of 1, the unknown TLV is readvertised along with the LDP message. This bit is not used by the hello parameters TLV and is set to a value of 0.

The remaining bits in this field represent the type code of the TLV. The hello parameters TLV uses a constant value of 0x0400 in this field.

Hello TLV Length (2 octets) This field displays the length, in bytes, of the remaining fields in the TLV. The hello parameters TLV uses a constant value of 4 in this field.

Hold Time (2 octets) This field displays the hold time, in seconds, requested by the advertising router. Each set of LDP neighbors negotiates a hold time for their relationship based on the lower value proposed by either neighbor. The JUNOS software uses a hold time of 15 seconds, by default, for all Hello messages.

Flags and Reserved (2 octets) The first 2 bits in this field, the T and R bits, are designated as flags and are discussed next. The remaining bits in the field are reserved and are set to a constant value of 0.

T Bit The first flag bit, the T bit, encodes the form of hello that the message represents. When the bit is set to a value of 1, the Hello message is a *targeted Hello*. This means that the two LDP neighbors are not directly connected across a network link. A value of 0 for this flag bit means that the Hello message is a *link Hello* and that the neighbors are directly connected.

R Bit This second flag bit, the R bit, is used when the Hello message is a targeted Hello. When it is set to a value of 1, the local router is requesting that its neighbor respond with its own targeted Hello message. A value of 0 for the flag makes no such request.

Optional TLV Type (2 octets) This field contains the type code for any optional TLV used in the message. The JUNOS software uses both an IPv4 transport address TLV (type code of 0x0401) and a configuration sequence number TLV (type code of 0x0402). Both of these TLVs set the U and F flag bits to 0 values.

The use of two TLVs means that the optional type, length, and value fields are repeated multiple times in a single Hello message. The transport address TLV informs the receiving router which address to use when establishing its LDP session with the local router. The configuration sequence number TLV is used by the sending router to alert any receiving routers that configuration changes have been made to the local router.

Optional TLV Length (2 octets) This field displays the length, in bytes, of the remaining optional TLV fields. Both the IPv4 transport address and configuration sequence number TLVs place a constant value of 4 in this field.

Optional TLV Value (Variable) This variable-length field contains the data carried by each optional TLV. The IPv4 transport address TLV places a 32-bit IP address in this field. By default, the JUNOS software uses the loopback address of the advertising router as the transport address.

The configuration sequence number TLV places a value of 1 in this field when LDP is first enabled on the local router. Each committed configuration change prompts the local router to increment this value by 1.

FIGURE 7.20 The LDP Hello message

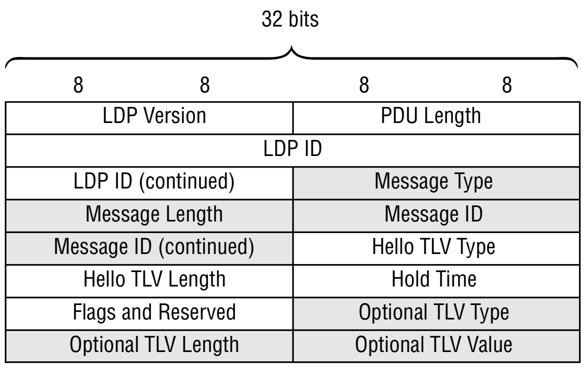
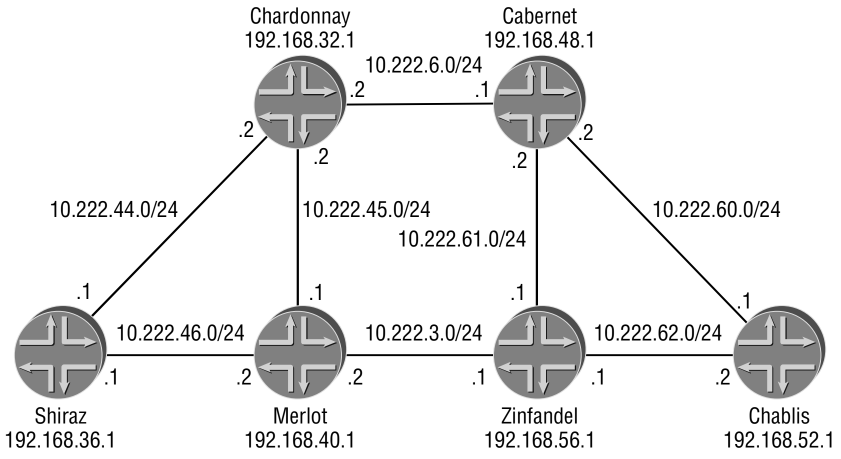


Figure 7.21 shows a sample network using LDP for advertising labels and establishing LSPs. Each of the six routers uses OSPF as the IGP, and each has enabled LDP on all of its operational interfaces. On the Chablis router, we see the advertisement of Hello messages to both Zinfandel and Cabernet in the output of a `traceoptions` file:

```
May 13 13:53:19 LDP sent UDP PDU 10.222.60.1 -> 224.0.0.2 (so-0/1/0.0)
May 13 13:53:19 ver 1, pkt len 42, PDU len 38, ID 192.168.52.1:0
May 13 13:53:19   Msg Hello (0x100), len 28, ID 1
May 13 13:53:19     TLV HelloParms (0x400), len 4
May 13 13:53:19       Hold time 15, flags <> (0x0)
May 13 13:53:19     TLV XportAddr (0x401), len 4
May 13 13:53:19       Address 192.168.52.1
May 13 13:53:19     TLV ConfSeq (0x402), len 4
May 13 13:53:19       Sequence 1

May 13 13:53:19 LDP sent UDP PDU 10.222.62.2 -> 224.0.0.2 (so-0/1/1.0)
May 13 13:53:19 ver 1, pkt len 42, PDU len 38, ID 192.168.52.1:0
May 13 13:53:19   Msg Hello (0x100), len 28, ID 2
May 13 13:53:19     TLV HelloParms (0x400), len 4
May 13 13:53:19       Hold time 15, flags <> (0x0)
May 13 13:53:19     TLV XportAddr (0x401), len 4
May 13 13:53:19       Address 192.168.52.1
May 13 13:53:19     TLV ConfSeq (0x402), len 4
May 13 13:53:19       Sequence 1
```

FIGURE 7.21 LDP sample network



The first critical piece of information we notice in the Hello message is contained in the LDP header—the LDP ID. This ID value defines the *label space* used by the local router as a combination of a unique identifier and the label allocation method used. The `ID 192.168.52.1:0` router output tells us that Chablis is using its router ID of 192.168.52.1 as its unique identifier. The zero notation at the end of the LDP ID means that Chablis is allocating labels in a per-node fashion, as opposed to a per-interface fashion, which is the JUNOS software default allocation method. This notion of a label space is used when the LDP neighbors form a protocol session between themselves. Only a single session is established for each label space, regardless of the number of neighbor relationships. A second important piece of information is needed for forming the protocol session, and it is also contained in the Hello message. The IPv4 transport address TLV informs the receiving LDP router of the address to be used for the session establishment.

NOTE The absence of the IPv4 transport address TLV in a Hello message means that the source address of the message should be used as the transport address.

After each neighbor receives Hello messages from one another, a neighbor relationship is formed between those routers. The `show ldp neighbor` output on the Chablis router shows both the Zinfandel (10.222.62.1) and Cabernet (10.222.60.2) routers as active neighbors:

```
user@Chablis> show ldp neighbor
Address            Interface          Label space ID         Hold time
10.222.60.2        so-0/1/0.0         192.168.48.1:0           12
10.222.62.1        so-0/1/1.0         192.168.56.1:0           12
```

Establishing LDP Sessions

Once two neighboring LDP routers know each other's label space and transport address via the Hello messages, an *LDP session* is established between those neighbors. This session is used for the advertisement of IPv4 interface addresses, labels, and reachable prefixes across an LSP. For reliable transport of this information, the session is established across a TCP connection between the routers. This TCP connection also uses the LDP well-known port number of 646 and is initiated by the router with the higher router ID, also known as the *active node*. Once the neighboring routers can communicate over the TCP transport connection, the active node generates and sends an *LDP initialization message* to its peer, the *passive node*. This initialization message contains basic information required to establish an LDP session between the neighbors, such as keepalive times and the LDP ID the session is connecting to. The JUNOS software also includes an optional TLV for support of graceful restart capabilities. The format of the LDP initialization message is shown in Figure 7.22; the fields are as follows:

LDP Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to constant value of 1.

PDU Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the PDU.

LDP ID (6 octets) This field contains the LDP ID of the sending router.

Message Type (2 octets) The U bit is set to a value of 0 for all initialization messages. The remaining bits in this field represent the type of message contained in the packet. Initialization messages place a constant value of 0x0200 in this field.

Message Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the message.

Message ID (4 octets) This field contains a 32-bit value generated by the advertising router that uniquely identifies this particular message.

Session TLV Type (2 octets) Both the U and F bits are set to 0 within the session parameters TLV. The remaining bits in this field represent the type code of the TLV, which is set to a constant value of 0x0500 for the session parameters TLV.

Session TLV Length (2 octets) This field displays the length, in bytes, of the remaining fields in the TLV. The session parameters TLV places a constant value of 14 in this field.

Protocol Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to constant value of 1.

Hold Time (2 octets) This field displays the amount of time, in seconds, requested by the advertising router as a hold time. Each set of LDP neighbors negotiates a hold time for their relationship based on the lower value proposed by either neighbor. The JUNOS software uses a hold time of 30 seconds, by default. However, the loss of a valid neighbor (using a 15-second hold time) results in the loss of the LDP session as well.

Flags and Reserved (1 octet) The first 2 bits in this field, the A and D bits, are designated as flags and are discussed next. The remaining bits in the field are reserved and are set to a constant value of 0.

A Bit The first flag bit, the A bit, encodes the method by which the local router advertises MPLS labels to peers. When the bit is cleared and set to 0, the sending router is using *Downstream Unsolicited* for advertising labels to peers. This means that the local router may send a label value upstream at any time after the LDP session is established. Setting the A bit to a value of 1 designates that the sending router is using *Downstream on Demand* for label advertisement. This requires the upstream router to explicitly request a label mapping before a label is advertised. The JUNOS software uses Downstream Unsolicited, by default, for all LDP sessions.

D Bit This second flag bit, the D bit, is used when loop detection is being used based on advertised path vector TLVs encoded in all LDP messages. A value of 0 signals that loop detection is not enabled on the sending router. A value of 1, on the other hand, means that loop detection is enabled on the sending router. The JUNOS software sets this flag to a 0 value because loop prevention is accomplished through means outside of LDP.

Path Vector Limit (1 octet) This field is used to set a limit on the number of hops a message may traverse before a loop is assumed. When loop detection is not used in the network, this field is not used and is set to a constant value of 0x00.

Maximum PDU Length (2 octets) This field allows the LDP neighbors to negotiate the maximum PDU length each may transmit across the network link connecting them. This value is a negotiable item, and both neighbors use the lower advertised value. The JUNOS software places a value of 4096 in this field by default.

Receiver LDP ID (6 octets) This field contains the LDP ID of the router that is receiving the initialization message. When combined with the LDP ID of the sending router, in the PDU header, this value allows the receiving router to associate the LDP session establishment with an active LDP neighbor.

Fault Tolerant TLV Type (2 octets) The JUNOS software places the Fault Tolerant TLV in all initialization messages, by default, for support of graceful restart. The U bit is set to a value of 1, while the F bit is left clear. This allows receiving routers to silently ignore the TLV but not forward it along to any other LDP neighbors.

The remaining bits in this field display the type code of the TLV—0x0503. When combined with the settings of the U and F flags, the final TLV type appears as 0x8503.

Fault Tolerant TLV Length (2 octets) This field displays the length, in bytes, of the remaining TLV fields. A constant value of 12 is placed in this field.

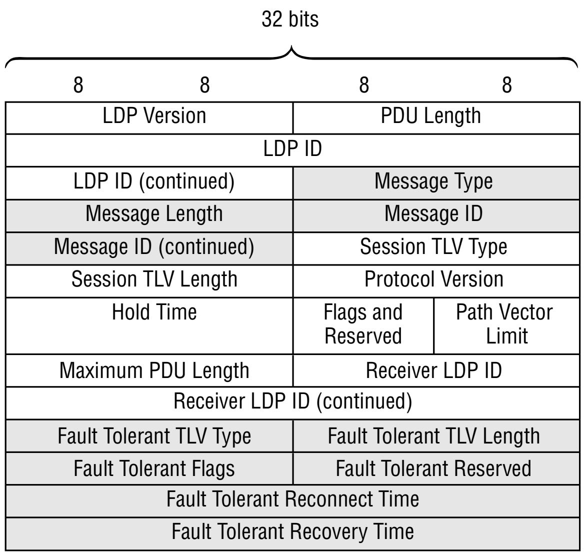
Fault Tolerant Flags (2 octets) This field contains numerous flags that describe the state of the fault-tolerant restart session. For a full explanation of each flag, please refer to RFC 3479, "Fault Tolerance for the Label Distribution Protocol."

Reserved (2 octets) This field is not used and is set to a constant value of 0x0000.

Fault Tolerant Reconnect Time (4 octets) This field displays the amount of time, in milliseconds, that the sending router maintains control state associated with the restarting router.

Fault Tolerant Recovery Time (4 octets) This field displays the maximum amount of time, in milliseconds, that the sending router uses for a restart timer. A value of 0 indicates that the sending router did not maintain the forwarding state throughout the restart event.

FIGURE 7.22 The LDP initialization message



We've seen that the Chablis router in Figure 7.21 has active neighbor relationships with both the Zinfandel and Cabernet routers. When we focus our attention on just the session establishment between Chablis and Cabernet, we find that the Chablis router is the active node for the session because its router ID is higher than Cabernet's. As such, Chablis generates an LDP initialization message and forwards it to Cabernet:

```
May 13 13:53:34 LDP sent TCP PDU 192.168.52.1 -> 192.168.48.1 (none)
May 13 13:53:34 ver 1, pkt len 52, PDU len 48, ID 192.168.52.1:0
May 13 13:53:34   Msg Initialization (0x200), len 38, ID 8
May 13 13:53:34     TLV SesParms (0x500), len 14
May 13 13:53:34       Ver 1, holdtime 30, flags <> (0x0)
May 13 13:53:34       vect_lim 0, max_pdu 4096, id 192.168.48.1:0
May 13 13:53:34     TLV GracefulRestartParms (0x8503), len 12
May 13 13:53:34       Reconnect time 0 ms, recovery time 0 ms
```

In addition to the negotiable values of hold time and maximum PDU, Chablis includes the LDP ID of Cabernet (192.168.48.1:0) in the message. This allows Cabernet to locate the appropriate neighbor relationship to associate the session with. Once it is located, Cabernet responds to Chablis with its own initialization message:

```
May 13 13:53:34 LDP rcvd TCP PDU 192.168.48.1 -> 192.168.52.1 (none)
May 13 13:53:34 ver 1, pkt len 52, PDU len 48, ID 192.168.48.1:0
May 13 13:53:34   Msg Initialization (0x200), len 38, ID 55
May 13 13:53:34     TLV SesParms (0x500), len 14
May 13 13:53:34       Ver 1, holdtime 30, flags <> (0x0)
May 13 13:53:34       vect_lim 0, max_pdu 4096, id 192.168.52.1:0
May 13 13:53:34     TLV GracefulRestartParms (0x8503), len 12
May 13 13:53:34       Reconnect time 0 ms, recovery time 0 ms
```

This exchange of messages allows the LDP session between the peers to fully establish itself. Cabernet now appears in the output of the `show ldp session` command:

```
user@Chablis> show ldp session
  Address           State       Connection     Hold time
192.168.48.1        Operational Open             20
192.168.56.1        Operational Open             28
```

The addition of the *`detail`* option allows us to view additional information pertaining to the established session:

```
user@Chablis> show ldp session detail 192.168.48.1
Address: 192.168.48.1, State: Operational, Connection: Open, Hold time: 27
  Session ID: 192.168.52.1:0--192.168.48.1:0
  Next keepalive in 7 seconds
  Active, Maximum PDU: 4096, Hold time: 30, Neighbor count: 1
  Keepalive interval: 10, Connect retry interval: 1
  Local address: 192.168.52.1, Remote address: 192.168.48.1
  Up for 22:45:00
  Local - Restart: disabled, Helper mode: enabled
  Remote - Restart: disabled, Helper mode: enabled
  Local maximum recovery time: 120000 msec
  Next-hop addresses received:
    so-0/1/0.0
    10.222.6.1
    192.168.48.1
    10.222.60.2
    10.222.61.2
```

The session between Chablis and Cabernet is currently `Operational` with a hold time of 30 seconds. Each router then calculates the timer used to send keepalive messages between the peers to maintain the session. The negotiated hold time of 30 seconds is divided by three, which results in a 10-second keepalive timer. The receipt of any LDP message within this timer window resets it to the maximum value. When no other messages have been transmitted within the keepalive time, the local router generates an *LDP keepalive message* and sends it to the remote peer. Figure 7.23 shows the format of the keepalive message, which includes the following fields:

LDP Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to a constant value of 1.

PDU Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the PDU. It is set to a constant value of 14.

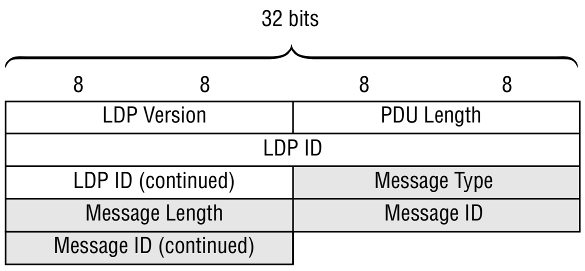
LDP ID (6 octets) This field contains the LDP ID of the sending router.

Message Type (2 octets) The U bit is set to a value of 0 for all keepalive messages. The remaining bits in this field represent the type of message contained in the packet. Keepalive messages place a constant value of 0x0201 in this field.

Message Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the message. It is set to a constant value of 4.

Message ID (4 octets) This field contains a 32-bit value generated by the advertising router that uniquely identifies this particular message.

FIGURE 7.23 The LDP keepalive message



Exchanging Information Across a Session

Once two LDP peers establish a session between themselves, they begin advertising network knowledge across that session. This information consists of local interface addresses for each peer as well as label values for forwarding MPLS packets through the network. Each set of information uses its own message formats and contains unique information. Let's examine each in further detail.

Advertising Interface Addresses

The first set of information transmitted between two LDP peers is the IPv4 interface address for all LDP operational interfaces on the local router. This allows the receiving router to associate future label advertisements with a physical next-hop address for the local router. These addresses are advertised in an *LDP address message*, which is displayed in Figure 7.24. The fields of the message include:

LDP Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to a constant value of 1.

PDU Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the PDU.

LDP ID (6 octets) This field contains the LDP ID of the sending router.

Message Type (2 octets) The U bit is set to a value of 0 for all address messages. The remaining bits in this field represent the type of message contained in the packet. Address messages place a constant value of 0x0300 in this field.

Message Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the message.

Message ID (4 octets) This field contains a 32-bit value generated by the advertising router that uniquely identifies this particular message.

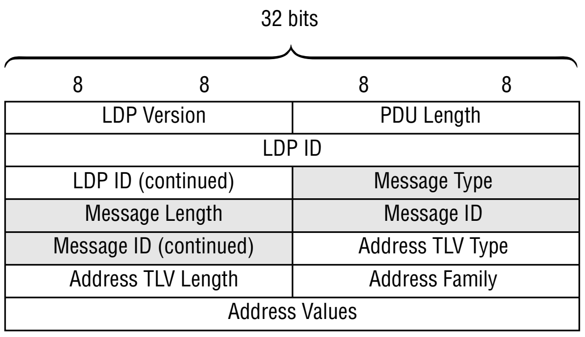
Address TLV Type (2 octets) Both the U and F bits are set to 0 within the address list TLV. The remaining bits in this field represent the type code of the TLV, which is set to a constant value of 0x0101 for the address list TLV.

Address TLV Length (2 octets) This field displays the length, in bytes, of the remaining fields in the TLV.

Address Family (2 octets) This field displays the type of addresses contained in the TLV. For IPv4 interface addresses, a constant value of 1 is placed in this field.

Address Values (Variable) This variable-length field displays the interface addresses being advertised by the local router. Each address is encoded as a 32-bit value.

FIGURE 7.24 LDP address message



The LDP session between the Chablis and Cabernet routers in Figure 7.21 can be used to examine the advertisement of addresses associated with those peers. Chablis currently has three addresses that are associated with LDP interfaces. We see those address values transmitted in an address message to Cabernet:

```
May 13 13:53:34 LDP sent TCP PDU 192.168.52.1 -> 192.168.48.1 (none)
May 13 13:53:34 ver 1, pkt len 36, PDU len 32, ID 192.168.52.1:0
May 13 13:53:34   Msg Address (0x300), len 22, ID 10
May 13 13:53:34     TLV AddrList (0x101), len 14
May 13 13:53:34      Address list, family 1
May 13 13:53:34        192.168.52.1
May 13 13:53:34        10.222.60.1
May 13 13:53:34        10.222.62.2
```

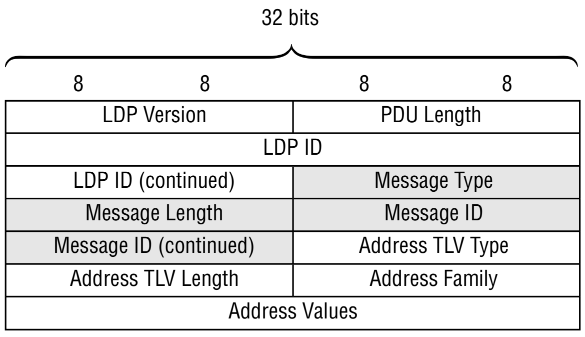
These address values represent the loopback interface of Chablis as well as its connections to Zinfandel and Cabernet. If Chablis has a requirement for advertising additional interface addresses in the future, it simply generates a new address message containing the new address and sends it to its peers.

From the perspective of the Cabernet router, there are four interface addresses that it needs to send to Chablis. These include the physical network links to Chardonnay, Zinfandel, and Chablis as well as its own local loopback interface. The address message containing this information looks like this:

```
May 13 13:53:34 LDP rcvd TCP PDU 192.168.48.1 -> 192.168.52.1 (none)
May 13 13:53:34 ver 1, pkt len 40, PDU len 36, ID 192.168.48.1:0
May 13 13:53:34  Msg Address (0x300), len 26, ID 57
May 13 13:53:34    TLV AddrList (0x101), len 18
May 13 13:53:34      Address list, family 1
May 13 13:53:34        10.222.6.1
May 13 13:53:34        192.168.48.1
May 13 13:53:34        10.222.60.2
May 13 13:53:34        10.222.61.2
```

If either of these peers disables LDP on an interface, or the interface is no longer operational, the previously advertised address must be removed from the peer's database. This is accomplished with an LDP address withdraw message, which is displayed in Figure 7.25. The fields of this message, as well as their meanings, are identical to those described for the address message. The only difference between the two LDP messages is the message type code value, which is set to 0x0301 for an address withdraw message.

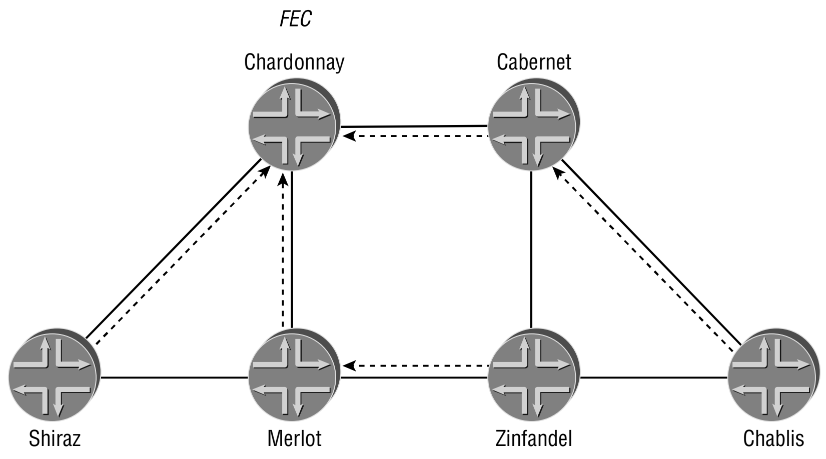
FIGURE 7.25 The LDP address withdraw message



Advertising Label Values

The main purpose for using LDP in a network is the establishment of LSPs for forwarding data traffic using MPLS labels. As such, it is critical that we understand how LDP advertises label values to each router in the network. Before discussing this advertisement in detail, however, we first need to examine a concept known as a *forwarding equivalence class* (FEC). In simple terms, a FEC is a prefix that is mapped by an egress router to an LSP. More specifically, a FEC represents a flow of IP packets through an MPLS network where each packet in the flow is processed identically and forward across the same physical path. For the purposes of our discussion, we'll assume that a FEC represents an IP prefix reachable through the egress router by means of an IP routing table lookup. The egress router then advertises this information upstream to all LDP peers with a specific label value assigned to it. These peers become an ingress router for an LSP that terminates at the egress router. In addition, these peers readvertise the FEC further upstream with a label value allocated locally, making them a transit router in the network. The peers that receive this readvertisement of the FEC then become ingress routers themselves. In this manner, each router in the entire LDP network has a method of forwarding MPLS packets to the FEC advertised by the single egress router. Figure 7.26 shows a FEC advertised by the Chardonnay router and the MPLS forwarding path used by each other router in the network to reach this advertised prefix.

FIGURE 7.26 Forwarding equivalence class example



A Juniper Networks router, by default, advertises only its 32-bit loopback address as a FEC. This allows all LDP routers in the network to establish an LSP to the loopback address of every other router in the network, creating a full mesh of LSPs. These MPLS-reachable addresses are placed into the `inet.3` routing table, where the Border Gateway Protocol (BGP) recursive lookup may locate them. The end result is the forwarding of BGP transit traffic across the network using label-switched paths established by LDP.

Every prefix advertised as a FEC is associated with a label value allocated by each router along the path of the LSP. This allows label advertisements to proceed in an upstream manner from the egress router to each ingress router without an explicit request for a label. The advertisement of a FEC and its label is accomplished with an *LDP label mapping message*. An individual message may contain a single label and FEC pair or multiple pairs. Figure 7.27 shows the format of the label mapping message, which includes the following fields:

LDP Version (2 octets) This field displays the current version of LDP being used by the sending router. It is set to a constant value of 1.

PDU Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the PDU.

LDP ID (6 octets) This field contains the LDP ID of the sending router.

Message Type (2 octets) The U bit is set to a value of 0 for all label mapping messages. The remaining bits in this field represent the type of message contained in the packet. Label mapping messages place a constant value of 0x0400 in this field.

Message Length (2 octets) This field displays the total length, in bytes, of the remaining fields contained in the message.

Message ID (4 octets) This field contains a 32-bit value generated by the advertising router that uniquely identifies this particular message.

FEC TLV Type (2 octets) Both the U and F bits are set to 0 within the FEC TLV. The remaining bits in this field represent the type code of the TLV, which is set to a constant value of 0x0100 for the FEC TLV.

FEC TLV Length (2 octets) This field displays the length, in bytes, of the remaining fields in the TLV.

FEC Element Type (1 octet) This field displays the specific form of the information contained in the FEC TLV. The three defined element types are

> **Wildcard** The wildcard FEC element, type code 0x01, is used only to remove previously advertised FEC and label values. It contains no information beyond the FEC element type field.
>
> **Prefix** The prefix FEC element, type code 0x02, is used by default for all addresses advertised by the JUNOS software. It is used to advertise and remove FEC prefixes in the network.
>
> **Host Address** The host address FEC element, type code 0x03, is used to advertise a complete and individual host address into the network.

Address Family (2 octets) This field displays the type of address contained in the FEC TLV. For IPv4 prefixes, a constant value of 1 is placed in this field.

Prefix Length (1 octet) This field displays the length of the prefix advertised in the FEC.

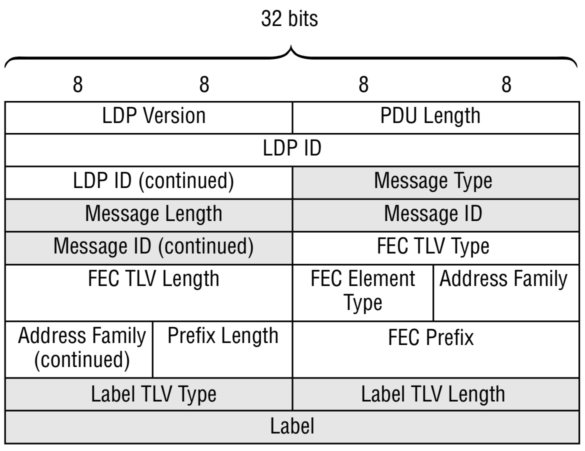
FEC Prefix (Variable) This variable-length field contains the address advertised by the egress router as a FEC.

Label TLV Type (2 octets) Both the U and F bits are set to 0 within the label TLV. The remaining bits in this field represent the type code of the TLV, which is set to a constant value of 0x0200 for the label TLV.

Label TLV Length (2 octets) This field displays the length, in bytes, of the remaining fields in the TLV. A constant value of 4 is placed in this field.

Label (4 octets) This field contains the label value associated with the FEC contained in the FEC TLV.

FIGURE 7.27 The LDP label mapping message



Using the LDP session between Chablis and Cabernet in Figure 7.21 as a guide, we see that the Chablis router advertises its loopback address of 192.168.52.1 /32 as a FEC to Cabernet. The label value associated with this prefix is 3, which signals Cabernet to perform PHP when forwarding traffic to this address. The label mapping message containing this information is seen in the output of a `traceoptions` file on the Chablis router:

```
May 13 13:53:34 LDP sent TCP PDU 192.168.52.1 -> 192.168.48.1 (none)
May 13 13:53:34 ver 1, pkt len 38, PDU len 34, ID 192.168.52.1:0
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 11
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.52.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 3
```

The Cabernet router receives this label mapping message and places it into a local database structure where it is associated with the LDP session it arrived on. Cabernet then allocates a label value from its local set of available labels and readvertises the 192.168.52.1 /32 FEC to all of its LDP peers. This includes the Chablis router, as we see in this router output:

```
May 13 13:53:34 LDP rcvd TCP PDU 192.168.48.1 -> 192.168.52.1 (none)
May 13 13:53:34 ver 1, pkt len 38, PDU len 34, ID 192.168.48.1:0
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 64
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.52.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 104352
```

In addition to its local FEC of 192.168.48.1 /32, the Cabernet router has received FEC advertisements for each of the other routers in the network—Zinfandel, Merlot, Shiraz, and Chardonnay. After allocating labels for each of the FECs, Cabernet advertises these prefixes to Chablis. Each individual FEC and label pair is encoded in a separate label mapping message contained in a single LDP PDU:

```
May 13 13:53:34 LDP rcvd TCP PDU 192.168.48.1 -> 192.168.52.1 (none)
May 13 13:53:34 ver 1, pkt len 150, PDU len 146, ID 192.168.48.1:0
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 58
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.48.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 3
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 59
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.32.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 104288
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 60
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.36.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 104304
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 61
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.40.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 104320
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 62
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.56.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 104336
```

As we saw earlier, Chablis receives these messages and stores them locally in a database. It then allocates labels for each of the FECs and advertises them to all LDP peers, including Cabernet:

```
May 13 13:53:34 LDP sent TCP PDU 192.168.52.1 -> 192.168.48.1 (none)
May 13 13:53:34 ver 1, pkt len 150, PDU len 146, ID 192.168.52.1:0
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 12
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.48.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 100000
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 13
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.32.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 100016
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 14
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.36.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 100032
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 15
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.40.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 100048
May 13 13:53:34   Msg LabelMap (0x400), len 24, ID 16
May 13 13:53:34     TLV FEC (0x100), len 8
May 13 13:53:34       Prefix, family 1, 192.168.56.1/32
May 13 13:53:34     TLV Label (0x200), len 4
May 13 13:53:34       Label 100064
```

The end result of this flooding is that each LDP router in the network receives a label from each LDP peer for every possible advertised FEC. This information is stored locally on each router within an *LDP database* associated with each peering session. The database displays the label-to-FEC mappings it has received from that peer, as well as the label-to-FEC mappings it advertised to the peer. For the Chablis-Cabernet peering session, we find the following output on the Chablis router:

```
user@Chablis> show ldp database session 192.168.48.1
Input label database, 192.168.52.1:0--192.168.48.1:0
  Label     Prefix
 104288     192.168.32.1/32
 104304     192.168.36.1/32
 104320     192.168.40.1/32
      3     192.168.48.1/32
 104352     192.168.52.1/32
 104336     192.168.56.1/32
```

```
Output label database, 192.168.52.1:0--192.168.48.1:0
  Label     Prefix
 100016     192.168.32.1/32
 100032     192.168.36.1/32
 100048     192.168.40.1/32
 100000     192.168.48.1/32
      3     192.168.52.1/32
 100064     192.168.56.1/32
```

At this point in our discussion, you might be worried about forwarding loops in the network. After all, if every router is going to advertise a label for every possible FEC, what prevents Chablis from forwarding a packet to Cabernet, which would forward it to Zinfandel, which would send it back to Chablis? The answer is quite simple really: LDP relies on the loop-prevention mechanisms of the network's IGP. The address of each received FEC is compared against the routing table of the router to verify that it currently has a valid physical next hop. This next-hop interface is then correlated to an LDP peering session based on the interface addresses received in the LDP address messages. Once the appropriate peering session is located, the local router takes the label value advertised over that session and places it, along with the FEC, in the `inet.3` routing table. The end result of this process is that the LSP from each ingress router to each egress router follows the IGP shortest path through the network.

NOTE This method of loop prevention requires that each address advertised in a FEC be reachable via the IGP before being readvertised across other LDP sessions.

We can verify this process by examining information on the Chablis router a bit closer. The current set of loopback addresses contained in the routing table looks like this:

```
user@Chablis> show route 192.168/16 terse protocol ospf

inet.0: 21 destinations, 23 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 192.168.32.1/32    O  10          2            >so-0/1/0.0
* 192.168.36.1/32    O  10          3            >so-0/1/0.0
                                                  so-0/1/1.0
* 192.168.40.1/32    O  10          2            >so-0/1/1.0
* 192.168.48.1/32    O  10          1            >so-0/1/0.0
* 192.168.56.1/32    O  10          1            >so-0/1/1.0
```

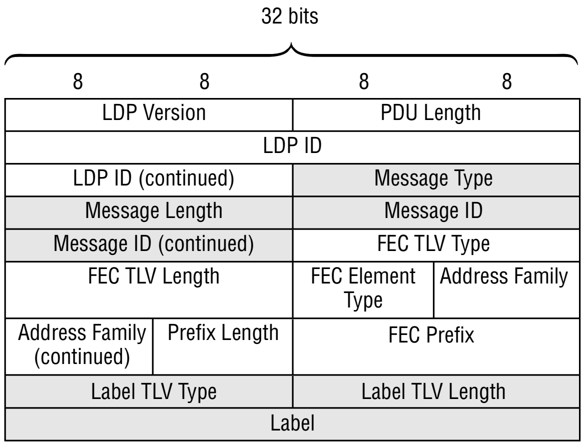
When we compare this output to the information contained in the `inet.3` routing table, we see that Chablis uses the same outgoing interface to reach each of the advertised FECs:

```
user@Chablis> show route table inet.3 terse

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 192.168.32.1/32    L   9          1            >so-0/1/0.0
* 192.168.36.1/32    L   9          1             so-0/1/0.0
                                                 >so-0/1/1.0
* 192.168.40.1/32    L   9          1            >so-0/1/1.0
* 192.168.48.1/32    L   9          1            >so-0/1/0.0
* 192.168.56.1/32    L   9          1            >so-0/1/1.0
```

The exception in this example is the 192.168.36.1 /32 route. This route has two equal-cost loop-free paths through the network. This allows each routing table to make its own selection for placement in the forwarding table. In our sample output, each table has selected a separate interface.

When an LDP router needs to remove a label and FEC that it had previously announced to a peer, it sends an *LDP label withdraw message* to that peer. This message is displayed in Figure 7.28, and it uses the same fields as the label mapping message. The main difference between the two messages is the type code used for the label withdraw message—0x0402.

FIGURE 7.28 The LDP label withdraw message



Using LDP through an RSVP Network

Some network administrators prefer using LDP for the establishment of LSPs but sometimes find a requirement to traffic-engineer their MPLS traffic. To address this situation, the JUNOS software permits the establishment of an LDP session across an RSVP network by tunneling the LDP traffic within RSVP-based LSPs. This type of network design is referred to as *LDP tunneling* and requires two RSVP signaled label-switched paths to exist between the LDP neighbors—one in each direction.

FIGURE 7.29 LDP tunneling

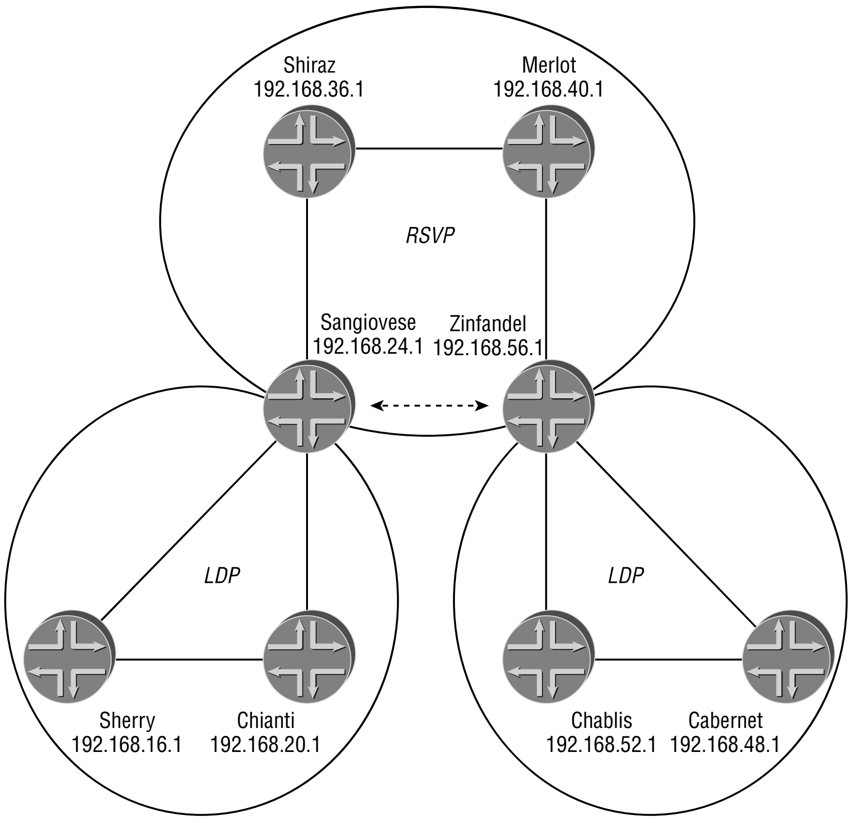


Figure 7.29 shows a network with two physically separated groups of LDP routers. The Sherry, Chianti, and Sangiovese routers are all LDP neighbors, as are the Chablis, Cabernet, and Zinfandel routers. We can verify the currently established LDP sessions in the network on the Sangiovese and Zinfandel routers:

```
user@Sangiovese> show ldp session
  Address           State       Connection     Hold time
```

```
192.168.16.1       Operational  Open            26
192.168.20.1       Operational  Open            26


user@Zinfandel> show ldp session
  Address          State        Connection     Hold time
192.168.48.1       Operational  Open            25
192.168.52.1       Operational  Open            29
```

The administrators of the network would like to forward user traffic from Cabernet to Sherry using MPLS labels. While this is easily accomplished by enabling LDP on the Shiraz and Merlot routers, the network requirement also calls for the engineering of traffic flows across the portion of the network already running RSVP. The solution to this administrative decision is to establish an LDP session between Sangiovese and Zinfandel using LDP tunneling. The first step in enabling this session is the creation of RSVP-based LSPs between the two remote LDP routers. These LSPs are currently operational across the RSVP core of the network:

```
user@Zinfandel> show mpls lsp
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.24.1    192.168.56.1    Up     0                  *     from-Zinfandel
Total 1 displayed, Up 1, Down 0

Egress LSP: 1 sessions
To              From            State Rt Style Labelin Labelout LSPname
192.168.56.1    192.168.24.1    Up     0  1 FF       3        - from-Sangiovese
Total 1 displayed, Up 1, Down 0

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

The second step in forming an LDP session between the two remote routers is the application of the `ldp-tunneling` command to the RSVP-based LSPs:

```
[edit protocols mpls]
user@Zinfandel# set label-switched-path from-Zinfandel ldp-tunneling


[edit protocols mpls]
user@Sangiovese# set label-switched-path from-Sangiovese ldp-tunneling
```

Once we commit our configurations, both Sangiovese and Zinfandel begin sending LDP targeted Hello messages to each other through the LSPs. These messages are sent to the address of the RSVP egress router, as seen on the Sangiovese router:

```
May 15 10:54:55 LDP sent UDP PDU 192.168.24.1 -> 192.168.56.1 (lo0.0)
May 15 10:54:55 ver 1, pkt len 42, PDU len 38, ID 192.168.24.1:0
May 15 10:54:55   Msg Hello (0x100), len 28, ID 768
May 15 10:54:55     TLV HelloParms (0x400), len 4
May 15 10:54:55       Hold time 15, flags <Targ ReqTarg> (0xc000)
May 15 10:54:55     TLV XportAddr (0x401), len 4
May 15 10:54:55       Address 192.168.24.1
May 15 10:54:55     TLV ConfSeq (0x402), len 4
May 15 10:54:55       Sequence 5
```

The flags set in the Hello message mark this as a targeted Hello and request that the receiving router respond with its own targeted Hello message. Since Zinfandel has an established RSVP LSP that egresses at Sangiovese and the LSP is configured for `ldp-tunneling`, a targeted Hello is returned by Zinfandel:

```
May 15 10:54:57 LDP rcvd UDP PDU 192.168.56.1 -> 192.168.24.1 (lo0.0)
May 15 10:54:57 ver 1, pkt len 42, PDU len 38, ID 192.168.56.1:0
May 15 10:54:57   Msg Hello (0x100), len 28, ID 166073
May 15 10:54:57     TLV HelloParms (0x400), len 4
May 15 10:54:57       Hold time 15, flags <Targ ReqTarg> (0xc000)
May 15 10:54:57     TLV XportAddr (0x401), len 4
May 15 10:54:57       Address 192.168.56.1
May 15 10:54:57     TLV ConfSeq (0x402), len 4
May 15 10:54:57       Sequence 5
```

At this point, the LDP neighbor relationship is formed and the routers exchange LDP initialization messages. The end result of the process is an LDP session between the routers and the exchange of label information:

```
user@Sangiovese> show ldp session
  Address           State        Connection     Hold time
192.168.16.1        Operational  Open             25
192.168.20.1        Operational  Open             25
192.168.56.1        Operational  Open             24


user@Sangiovese> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.168.56.1       lo0.0              192.168.56.1:0           13
10.222.28.1        at-0/1/0.0         192.168.16.1:0           12
```

```
10.222.30.2       so-0/2/0.0        192.168.20.1:0          11

user@Sangiovese> show ldp database session 192.168.56.1
Input label database, 192.168.24.1:0--192.168.56.1:0
  Label     Prefix
 108480     192.168.16.1/32
 108496     192.168.20.1/32
 108464     192.168.24.1/32
 108432     192.168.48.1/32
 108448     192.168.52.1/32
      3     192.168.56.1/32

Output label database, 192.168.24.1:0--192.168.56.1:0
  Label     Prefix
 100352     192.168.16.1/32
 100368     192.168.20.1/32
      3     192.168.24.1/32
 100384     192.168.48.1/32
 100400     192.168.52.1/32
 100416     192.168.56.1/32
```

NOTE The use of the lo0.0 interface for the neighbor relationship across the RSVP LSP requires running LDP on the loopback interface of the peering routers.

Once the session between Sangiovese and Zinfandel is established, the Cabernet router receives a FEC for the loopback address of Sherry. This address, as well as the label mapping information from Zinfandel, is placed into the `inet.3` routing table:

```
user@Cabernet> show route table inet.3 192.168.16.1

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

192.168.16.1/32    *[LDP/9] 00:31:52, metric 1
                    > via so-0/1/0.0, Push 108480
```

This table entry allows Cabernet to push a label value of 108,480 onto a packet and forward it out its `so-0.1.0.0` interface to the Zinfandel router. Zinfandel should, in turn, forward the packet to its LDP peer of Sangiovese that advertised a label value of 100,352 for that FEC:

```
user@Zinfandel> show ldp database session 192.168.24.1
Input label database, 192.168.56.1:0--192.168.24.1:0
```

```
  Label     Prefix
 100352     192.168.16.1/32
 100368     192.168.20.1/32
      3     192.168.24.1/32
 100384     192.168.48.1/32
 100400     192.168.52.1/32
 100416     192.168.56.1/32

Output label database, 192.168.56.1:0--192.168.24.1:0
  Label     Prefix
 108480     192.168.16.1/32
 108496     192.168.20.1/32
 108464     192.168.24.1/32
 108432     192.168.48.1/32
 108448     192.168.52.1/32
      3     192.168.56.1/32
```

Under normal circumstances, Zinfandel would swap label 108,480 with 100,352 and forward the packet directly to Sangiovese. In this particular case, however, Sangiovese is reachable only across an RSVP LSP. This means that along with performing the swap operation so that Sangiovese receives the label it advertised for the FEC, the Zinfandel router must push an additional label value onto the stack. This second label value (100,304) was advertised by Merlot for the RSVP LSP that egresses at Sangiovese. This information is contained in the output of the `show rsvp session ingress` command on Zinfandel:

```
user@Zinfandel> show rsvp session ingress
Ingress RSVP: 1 sessions
To              From            State Rt Style Labelin Labelout LSPname
192.168.24.1    192.168.56.1    Up     0  1 FF       -   100304 from-Zinfandel
Total 1 displayed, Up 1, Down 0
```

This simultaneous swap and push operation on Zinfandel is visible by examining the `mpls.0` switching table using the *`detail`* option:

```
user@Zinfandel> show route table mpls.0 detail | find 108480
108480 (1 entry, 1 announced)
        *LDP    Preference: 9
                Next hop: via so-0/1/1.0 weight 1, selected
                Label-switched-path from-Zinfandel
                Label operation: Swap 100352, Push 100304(top)
                State: <Active Int>
                Age: 2  Metric: 1
                Task: LDP
                Announcement bits (1): 0-KRT
                AS path: I
                Prefixes bound to route: 192.168.16.1/32

108496 (1 entry, 1 announced)
        *LDP    Preference: 9
                Next hop: via so-0/1/1.0 weight 1, selected
                Label-switched-path from-Zinfandel
                Label operation: Swap 100368, Push 100304(top)
                State: <Active Int>
                Age: 2  Metric: 1
                Task: LDP
                Announcement bits (1): 0-KRT
                AS path: I
                Prefixes bound to route: 192.168.20.1/32
---(more)---[abort]
```

The top MPLS label in the stack is swapped by Merlot when it receives the data packet. It is then forwarded to the Shiraz router, which performs a label pop operation and forwards the remaining data to Sangiovese. The packet received by Sangiovese has an MPLS label assigned that Sangiovese advertised as a FEC to Zinfandel. This allows the Sangiovese router to recognize the label and perform a label operation on the received packet. In our example, this label operation is a pop because the next-hop router, Sherry, is the egress router and requested that Sangiovese perform PHP.

Summary

In this chapter, we examined the Resource Reservation Protocol (RSVP) and how to use it to establish label-switched paths. After a discussion of the various RSVP message types, we explored the RSVP objects themselves at great depth. Finally, we discussed the concept of an RSVP session and used `show` commands to prove the network was operating normally.

We concluded the chapter with a discussion of the Label Distribution Protocol (LDP). After examining how two routers become LDP neighbors, we described the establishment of an LDP session between those peers. Once the session was established, we saw the advertisement of interface addresses, forwarding equivalence classes, and labels between those peers. Finally, we established a session between two physically remote neighbors by tunneling LDP through an RSVP-based LSP.

Exam Essentials

Be able to explain how an RSVP-based path is established in an MPLS network. An MPLS network using RSVP for signaling label-switched paths uses `Path` and `Resv` messages for the establishment of the LSP. `Path` messages are transmitted downstream, whereas `Resv` messages are transmitted upstream. Each message enables routers to allocate and maintain protocol state associated with the path.

Know the RSVP message types used in an MPLS network for removing protocol state. RSVP routers remove protocol state from the network by transmitting `PathTear` and `ResvTear` messages. These messages traverse the LSP path in the same direction of their counterparts. `PathTear` messages are transmitted downstream, whereas `ResvTear` messages are sent upstream.

Be able to describe the RSVP objects used to route an LSP through the network and allocate label information. An RSVP-based LSP uses the Explicit Route object to formulate a path through an MPLS network. This allows the LSP to utilize links other than that specified by the IGP shortest path. The actual path used by the LSP is detailed in the Record Route object, which is also used for loop prevention during the LSP establishment phase. Finally, the MPLS labels used to forward traffic through the LSP are assigned using the Label Request and Label objects.

Be able to describe how RSVP sessions are identified in an MPLS network. Each established label-switched path belongs to a specific RSVP session in the network. This session is uniquely identified by information contained in the Sender-Template, Session, and Session Attribute objects. Specifically, the address of the egress router, the tunnel ID, and the LSP ID are used to identify the session.

Be able to describe how LDP forms a session between two peers. Two neighboring LDP routers first begin exchanging Hello messages with each other. These messages contain the label space advertised by the local router as well as the transport address to be used for the establishment of the session. Once each router determines the neighbor on its interface, the peer with the higher transport address becomes the active node. This active node sends initialization messages to the passive node to begin the session setup phase. Once the passive peer returns its own initialization message, the session becomes fully established.

Understand how address and label information is propagated in an LDP network. After two LDP routers form a session between themselves, they begin advertising their local interface addresses to the peer with address messages. This allows the receiving peer to associate a physical next hop with the established session. Once this exchange is completed, the peers advertise reachable prefixes and labels in a label mapping message. The prefixes advertised by each LDP router compose a FEC, which is readvertised throughout the LDP network.

Review Questions

1. Which RSVP object appears in both a `Path` and a `Resv` message?
   - **A.** Label
   - **B.** Label Request
   - **C.** Explicit Route
   - **D.** Record Route
2. Which RSVP object is used to signal all routers along the LSP to protect against downstream link and node failures?
   - **A.** Detour
   - **B.** Fast Reroute
   - **C.** Session Attribute
   - **D.** Sender-Template
3. Which RSVP reservation style is used by default within the JUNOS software?
   - **A.** Fixed filter
   - **B.** Wildcard filter
   - **C.** Shared explicit
   - **D.** Shared wildcard
4. Which RSVP message removes existing reservation state from routers along the path of an LSP?
   - **A.** `PathTear` sent to the ingress router
   - **B.** `ResvTear` sent to the ingress router
   - **C.** `PathTear` sent to the egress router
   - **D.** `ResvTear` sent to the egress router
5. What three RSVP objects are used to uniquely identify an RSVP session in the network?
   - **A.** Sender-Template
   - **B.** Sender-Tspec
   - **C.** Session
   - **D.** Session Attribute
6. What prompts each MPLS router to examine the contents of an RSVP `Path` message?
   - **A.** The destination address is the local router's interface address.
   - **B.** The Router Alert option is set in the IP packet header.
   - **C.** It is sent to a well-known TCP port number.
   - **D.** None of the above.

7. What RSVP object is used to avoid a forwarding loop along the path of an LSP?
   - **A.** Explicit Route
   - **B.** Record Route
   - **C.** Session Attribute
   - **D.** Session

8. What is the destination address and protocol information for an LDP Hello message?
   - **A.** Destined to 224.0.0.2 and UDP port 646
   - **B.** Destined to 224.0.0.2 and TCP port 646
   - **C.** Destined to 224.0.0.1 and UDP port 646
   - **D.** Destined to 224.0.0.1 and TCP port 646

9. What information does the JUNOS software include by default in the FEC advertised by each router?
   - **A.** All IP prefixes located on the local router
   - **B.** All interface addresses for the local router
   - **C.** Only subnets through which an LDP neighbor is located
   - **D.** Only the local router's loopback address

10. How does an LDP network prevent routing and forwarding loops?
    - **A.** Label information is advertised only between specific peers.
    - **B.** LDP routers perform a reverse path forwarding check for all packets.
    - **C.** The shortest-path first calculation is performed against the contents of the LDP database.
    - **D.** LDP routers use the IGP shortest-path information contained in the routing table.

Answers to Review Questions

1. D. Of the listed RSVP objects, only the Record Route object may appear in both a `Path` and a `Resv` message. The Explicit Route and Label Request objects appear only in `Path` messages, whereas the Label object appears only in `Resv` messages.
2. B. When included in a `Path` message, the Fast Reroute object informs each LSP of the ingress router's desire to protect the LSP against downstream failures. Each node along the LSP path then creates a detour around the downstream resource using a `Path` message with the Detour object.
3. A. The fixed filter (FF) reservation style is used by default on all RSVP LSPs created by the JUNOS software.
4. B. Existing reservation state is removed from the network through the use of the `ResvTear` message. This message type is always advertised in a hop-by-hop fashion to the ingress router.
5. A, C, D. The unique set of information contained in the Sender-Template, Session, and Session Attribute objects uniquely identifies an RSVP session in the network.
6. B. Within the IP packet header of the RSVP `Path` message, the Router Alert option is enabled. This allows each router along the path to examine the contents even though the destination address of the packet is the egress router's address.
7. B. When each RSVP router receives either a `Path` or a `Resv` message, it examines the Record Route object looking for one of its interface addresses. If it finds one, that means a forwarding loop is in place within the network and the LSP is not established.
8. A. All LDP Hello messages are transmitted to the well-known UDP port number of 646. In addition, all link Hellos are sent to the 224.0.0.2 multicast group address representing all routers on the subnet.
9. D. By default, only the loopback address of the local router is advertised as part of the FEC for that router.
10. D. Routing and forwarding loops are prevented in an LDP network by consulting the shortest-path information in the local routing table. The outgoing interface for a particular route is mapped to the label received from the LDP peer associated with that interface.

Advanced MPLS

JNCIS EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ **Identify the operational steps of the constrained shortest path first algorithm**
- ✓ **Describe the extensions to OSPF that support traffic engineering**
- ✓ **Describe the extensions to IS-IS that support traffic engineering**
- ✓ **Define the reason(s) for configuring the following LSP attributes: primary; secondary; administrative groups; adaptive settings; priority/preemption; optimization; forwarding adjacencies into IGPs; fast reroute; metric; time-to-live options; auto-bandwidth; explicit null**

In this chapter, we construct a link-state database specifically designed for engineering label-switched path (LSP) traffic flows. The JUNOS software uses this traffic engineering database (TED) to generate a Resource Reservation Protocol (RSVP) path through the network based on constraints you provide. We discuss the contents of the database and the algorithm used on those contents. We then examine some methods for protecting traffic flows in the Multiprotocol Label Switching (MPLS) network. This discussion centers on providing secondary backup paths across the domain as well as using fast reroute to protect traffic already using the LSP. We conclude the chapter with an exploration of various attributes that affect individual LSPs. We discuss some bandwidth reservation options as well as some methods for hiding the physical connectivity of your MPLS network. Finally, we investigate how user data traffic not associated with Border Gateway Protocol (BGP) can use an LSP for forwarding packets across the network.

Constrained Shortest Path First

The ability to engineer LSP traffic flows using RSVP is greatly enhanced through the use of an algorithm known as *Constrained Shortest Path First* (CSPF). This algorithm is a modified version of the SPF algorithm used within the link-state databases of Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS). It operates within a special database, the TED, constructed through extensions to the link-state protocols themselves. We discuss the basic use of the TED by RSVP, the extensions defined for the interior gateway protocols (IGPs), and the steps of the CSPF algorithm itself.

Using the Traffic Engineering Database

The topology of an RSVP network is populated into the TED by the link-state protocols. This basic network topology includes the devices capable of running MPLS as well as information on bandwidth availability, available priority values, and administrative controls—to name a few. When you define an LSP within the JUNOS software, you have the ability to supply constraints to the ingress router. These user-defined criteria are passed to the CSPF algorithm before it consults the information in the TED. The algorithm removes all network links from consideration that don't meet the constraints. It then places the remaining topology data into a temporary data structure. The shortest path between the ingress and egress routers is calculated by the algorithm using this subset of information. The result of the CSPF algorithm is formed into a strict-hop Explicit Route object (ERO) detailing each hop along the calculated path. The ERO is passed to the RSVP protocol process, where it is used for signaling and establishing the LSP in the network.

The creation of this ERO does not, however, prevent you from creating your own explicit route via a named path in the JUNOS software. All user-defined path information is passed to the CSPF algorithm as a requested constraint. After creating the subset of network links within the TED, the algorithm consults the information in the user-defined named path. The router runs a separate CSPF calculation to locate the path from the ingress router to the first node listed in the user's ERO. A second CSPF calculation is performed to find a path from the first node to the second node in the user's ERO. The router repeats this pattern of CSPF calculations until the egress router is reached. The results of the multiple computations are then combined into a single strict-hop ERO from the ingress to the egress.

The ability of CSPF to locate a viable network path for an LSP is greatly affected by the information in the TED, so let's investigate how the IGPs propagate this data.

OSPF Traffic Engineering Extensions

The OSPF protocol makes use of a *Type 10 Opaque LSA* for advertising traffic engineering information in a network. This LSA type has an area flooding scope, which limits the creation of the TED to a single OSPF area. This means that a large OSPF network with multiple areas will have multiple TEDs, one for each operational area.

Figure 8.1 displays the format of the Opaque LSA used by OSPF for advertising traffic engineering information. The fields of the LSA include the following:

Link-State Age (2 octets) The Link-State Age field displays the time, in seconds, since the LSA was first originated into the network. The age begins at the value 0 and increments to a value of 3600 (1 hour).

Options (1 octet) The local router advertises its capabilities in this field, which also appears in other OSPF packets. The router sets bit 6 in this field, indicating its support for traffic engineering.

Link-State Type (1 octet) This field displays the type of link-state advertisement (LSA). Opaque LSAs set this field to a constant value of 10.

Opaque Type (1 octet) The Opaque LSA specification defines the first octet of the link-state ID field as the Opaque Type field. When used to support traffic engineering, a constant value of 1 is placed in this field.

Reserved (1 octet) This field is not used and is set to a constant value of 0x00.

Instance (2 octets) The Instance field of a traffic engineering Opaque LSA is used by the local router to support multiple, separate LSAs. By default, the JUNOS software generates one LSA for the router itself as well as a separate LSA for each operational interface.

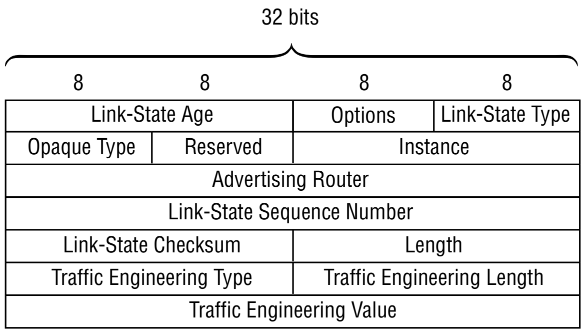
Advertising Router (4 octets) This field displays the router ID of the OSPF device that originated the LSA.

Link-State Sequence Number (4 octets) The sequence number field is a signed 32-bit value used to guarantee that each router in the area has the most recent version of the Opaque LSA.

Link-State Checksum (2 octets) This field displays a standard IP checksum for the entire LSA, excluding the Link-State Age field.

Length (2 octets) This field displays the length of the entire LSA, including the header fields.

FIGURE 8.1 Traffic engineering Opaque LSA format



Traffic Engineering Type (**2 octets**) The traffic engineering data is encoded within the Opaque LSA using a type, length, value (TLV) paradigm. Currently two TLV type values are defined. The Router Address TLV uses a type code of 1, whereas the Link TLV uses a type code of 2.

Traffic Engineering Length (**2 octets**) This field displays the length, in bytes, of the value portion of the TLV.

Traffic Engineering Value (**Variable**) This field displays the specific information contained within the TLV being advertised.

The Sherry router is running OSPF and has enabled the traffic engineering extensions by applying the `traffic-engineering` command at the global OSPF level. The configuration of Sherry currently looks like this:

```
user@Sherry> show configuration protocols ospf
traffic-engineering;
area 0.0.0.0 {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

This allows the Sherry router to generate traffic engineering Opaque LSAs, which it places into the local link-state database and floods to all its neighbors:

```
user@Sherry> show ospf database advertising-router 192.168.16.1

    OSPF link state database, area 0.0.0.0
 Type       ID               Adv Rtr          Seq      Age  Opt  Cksum  Len
Router  *192.168.16.1     192.168.16.1     0x80000370    55  0x2  0xbad8  72
Network *10.0.15.2        192.168.16.1     0x80000365   201  0x2  0xbcb3  32
Network *10.0.16.1        192.168.16.1     0x80000364   801  0x2  0x5524   32
OpaqArea*1.0.0.1          192.168.16.1     0x80000001    55  0x2  0xb0a9   28
OpaqArea*1.0.0.2          192.168.16.1     0x80000001    50  0x2  0x94ce  124
OpaqArea*1.0.0.3          192.168.16.1     0x80000001    50  0x2  0x7bf1  124
OpaqArea*1.0.0.4          192.168.16.1     0x80000001    50  0x2  0x1246  124
```

The LSA whose link-state ID is currently 1.0.0.1 contains the *Router Address TLV* representing the router itself. The TLV uses a type code value of 1 and has a constant length of 4 octets. The value contained in the Router Address TLV includes an IP address that is consistently reachable on the local router. The loopback address of the local router is used for this value, 192.168.16.1 in the case of the Sherry router:

```
user@Sherry> show ospf database lsa-id 1.0.0.1 detail

    OSPF link state database, area 0.0.0.0
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
OpaqArea*1.0.0.1          192.168.16.1     0x80000001    58  0x2  0xb0a9   28
  Area-opaque TE LSA
  RtrAddr (1), length 4:  192.168.16.1
```

Each of the remaining TLVs advertised by Sherry represents the operational links on the router, and each contains a single *Link TLV*. The value portion of the TLV contains multiple sub-TLVs to describe the traffic engineering capabilities of the interface. The type codes of the defined sub-TLVs are

Link Type The Link Type sub-TLV has a type code of 1 and a sub-TLV length of 1 octet. The value portion of the sub-TLV describes whether the link is a point-to-point or broadcast interface. Point-to-point interfaces use a value of 1 in the sub-TLV, whereas broadcast interfaces place a 2 in the value portion of the sub-TLV.

Link ID The Link ID sub-TLV uses a type code of 2 and describes the opposite end of the link. The sub-TLV has a length of 4 octets and contains an IP address in the value portion of the sub-TLV. For a point-to-point link, the router ID of the remote neighbor is encoded in the sub-TLV. The interface address of the designated router is placed in the sub-TLV for all broadcast interfaces.

Local Interface IP Address The use of the Local Interface IP Address sub-TLV is fairly straightforward—it contains the interface address of the local router for the link being described by the Opaque LSA. The sub-TLV uses a type code of 3 and has a length of 4 octets.

Remote Interface IP Address The Remote Interface IP Address sub-TLV contains the interface address of the remote router for the link being described by the Opaque LSA. The sub-TLV uses a type code of 4 and has a length of 4 octets. Point-to-point interfaces use the remote router's address in this field, whereas broadcast links use a value of 0.0.0.0. When the link is configured as an unnumbered interface, the first 2 octets of the value portion contain all zeros. The remaining 2 octets contain a unique interface index value.

Traffic Engineering Metric The Traffic Engineering Metric sub-TLV has a type code of 5 and a sub-TLV length of 4 octets. It contains the metric value of the interface that should be used by the CSPF algorithm. The JUNOS software places the same value in this sub-TLV as it places in the Router LSA of the local router.

Maximum Bandwidth The Maximum Bandwidth sub-TLV contains the true bandwidth of the local interface in bytes per second. It uses a type code of 6 and has a constant length of 4 octets.

Maximum Reservable Bandwidth The Maximum Reservable Bandwidth sub-TLV displays the total amount of bandwidth, in bytes per second, available for reservations by the RSVP process. This value may be larger than the actual bandwidth of the interface to account for oversubscription of the interface. The sub-TLV has a type code of 7 and a constant length of 4 octets.

Unreserved Bandwidth The type code of the Unreserved Bandwidth sub-TLV is 8. It displays the amount of bandwidth, in bytes per second, that is currently available for reservations on the local interface. This value is calculated for each of the eight priority levels used by the extensions to the RSVP specification. Each unreserved bandwidth value has a length of 4 octets, which results in the entire sub-TLV having a length of 32 octets.

Resource Class/Color The Resource Class/Color sub-TLV has a type code of 9 and a sub-TLV length of 4 octets. The value portion of the sub-TLV contains a 32-bit vector used to encode membership in various administrative groups. An individual interface may belong to multiple groups, in which case multiple bits are set within the bit vector.

The Sherry router is generating three Opaque LSAs to represent its three operational links in the traffic engineering domain. Here are the contents of one of the LSAs:

```
user@Sherry> show ospf database lsa-id 1.0.0.2 detail

    OSPF link state database, area 0.0.0.0
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
OpaqArea*1.0.0.2         192.168.16.1     0x80000001    53  0x2  0x94ce 124
  Area-opaque TE LSA
  Link (2), length 100:
    Linktype (1), length 1:
      2
    LinkID (2), length 4:
      10.222.28.2
    LocIfAdr (3), length 4:
      10.222.28.1
    RemIfAdr (4), length 4:
      0.0.0.0
    TEMetric (5), length 4:
      1
    MaxBW (6), length 4:
      100Mbps
    MaxRsvBW (7), length 4:
      100Mbps
    UnRsvBW (8), length 32:
        Priority 0, 100Mbps
        Priority 1, 100Mbps
        Priority 2, 100Mbps
        Priority 3, 100Mbps
        Priority 4, 100Mbps
        Priority 5, 100Mbps
        Priority 6, 100Mbps
        Priority 7, 100Mbps
    Color (9), length 4:
      0
```

The Opaque LSA describes a Fast Ethernet interface because the Link Type sub-TLV has a value of 2 and the maximum bandwidth displayed in sub-TLV 6 (`MaxBW`) shows 100Mbps. The Maximum Reservable Bandwidth (`MaxRsvBW`) and the Unreserved Bandwidth (`UnRsvBW`) sub-TLVs each are each reporting 100Mbps in their value portions, which means that the interface is not oversubscribed and no current bandwidth reservations are in place across the interface. The local IP address of the interface is 10.222.28.1, which is connected to the designated router address of 10.222.28.2. The connection to the designated router (DR) prompts the remote interface address to be set to all zeros. Finally, the Resource Class/Color sub-TLV displays a value of 0, which represents no current administrative group memberships.

IS-IS Traffic Engineering Extensions

IS-IS advertises traffic engineering information in the *Extended IS Reachability TLV*, whose type code is 22. The JUNOS software advertises this TLV by default for all operational IS-IS interfaces in a specific level. Its use within this context, as well as the format of the TLV, is discussed in Chapter 3, "Intermediate System to Intermediate System (IS-IS)." When a particular interface is also configured to support RSVP, the traffic engineering information is included in the Extended IS Reachability TLV in the form of sub-TLVs. The specific information contained in the sub-TLVs is as follows:

Administrative Group The Administrative Group sub-TLV has a type code of 3 and a length of 4 octets. The value portion of the sub-TLV contains a 32-bit vector used to encode membership in various administrative groups. An individual interface may belong to multiple groups, in which case multiple bits are set within the bit vector.

IPv4 Interface Address The IP address of the local router's interface is contained in the IPv4 Interface Address sub-TLV. It uses a type code of 6 and has a constant length of 4 octets.

IPv4 Neighbor Address The IP address of the remote router's interface is contained in the IPv4 Neighbor Address sub-TLV. It uses a type code of 8 and has a constant length of 4 octets. When the interface is operating in an unnumbered fashion, the neighbor's router ID is placed in this field.

Maximum Link Bandwidth The true bandwidth, in bytes per second, of the local router's interface is contained in the Maximum Link Bandwidth sub-TLV. It uses a type code of 9 and has a constant length of 4 octets.

Maximum Reservable Link Bandwidth The Maximum Reservable Bandwidth sub-TLV displays the total amount of bandwidth, in bytes per second, available for reservations by the RSVP process. This value may be larger than the actual bandwidth of the interface to account for oversubscription of the interface. The sub-TLV has a type code of 10 and a constant length of 4 octets.

Unreserved Bandwidth The Unreserved Bandwidth sub-TLV uses a type code of 11 and has a constant length of 32 octets. It displays the amount of bandwidth, in bytes per second, that is currently available for reservations on the local interface. This value is calculated for each of the eight priority levels used by the RSVP traffic engineering process. Each unreserved bandwidth value is reported in a 4-octet field.

Traffic Engineering Default Metric The Traffic Engineering Default Metric sub-TLV has a type code of 18 and a length of 3 octets. It is advertised only when the interface metric to be used for traffic engineering purposes differs from the interface metric used by the normal IS-IS routing process. The use of the `te-metric` command in the JUNOS software allows you to administratively set this metric value.

The Sherry router is running IS-IS and has enabled RSVP to support traffic engineering within the network. The configuration of the router currently looks like this:

```
user@Sherry> show configuration protocols
rsvp {
    interface all;
}
mpls {
    interface all;
}
isis {
    level 1 disable;
    level 2 wide-metrics-only;
    interface ge-0/2/0.0 {
        level 2 te-metric 30;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

This configuration allows Sherry to include traffic engineering information within the Extended IS Reachability TLV. We can view this data by examining the details of the link-state PDU generated by Sherry:

```
user@Sherry> show isis database Sherry.00-00 extensive | find TLV
  TLVs:
```

```
Area address: 49.1111 (3)
Speaks: IP
Speaks: IPv6
IP router id: 192.168.16.1
IP address: 192.168.16.1
Hostname: Sherry
IS extended neighbor: Sangiovese.00, Metric: default 10
  IP address: 10.222.28.1
  Neighbor's IP address: 10.222.28.2
  Current reservable bandwidth:
    Priority 0 : 155.52Mbps
    Priority 1 : 155.52Mbps
    Priority 2 : 155.52Mbps
    Priority 3 : 155.52Mbps
    Priority 4 : 155.52Mbps
    Priority 5 : 155.52Mbps
    Priority 6 : 155.52Mbps
    Priority 7 : 155.52Mbps
  Maximum reservable bandwidth: 155.52Mbps
  Maximum bandwidth: 155.52Mbps
  Administrative groups:  0  <none>
IS extended neighbor: Chablis.00, Metric: default 10
  IP address: 10.222.5.1
  Neighbor's IP address: 10.222.5.2
  Current reservable bandwidth:
    Priority 0 : 155.52Mbps
    Priority 1 : 155.52Mbps
    Priority 2 : 155.52Mbps
    Priority 3 : 155.52Mbps
    Priority 4 : 155.52Mbps
    Priority 5 : 155.52Mbps
    Priority 6 : 155.52Mbps
    Priority 7 : 155.52Mbps
  Maximum reservable bandwidth: 155.52Mbps
  Maximum bandwidth: 155.52Mbps
  Administrative groups:  0  <none>
IS extended neighbor: Chianti.02, Metric: default 10
  IP address: 10.222.29.1
  Traffic engineering metric: 30
  Current reservable bandwidth:
          Priority 0 : 1000Mbps
          Priority 1 : 1000Mbps
          Priority 2 : 1000Mbps
          Priority 3 : 1000Mbps
          Priority 4 : 1000Mbps
          Priority 5 : 1000Mbps
          Priority 6 : 1000Mbps
          Priority 7 : 1000Mbps
        Maximum reservable bandwidth: 1000Mbps
        Maximum bandwidth: 1000Mbps
        Administrative groups:  0  <none>
      IP extended prefix: 10.222.28.0/24 metric 10 up
      IP extended prefix: 10.222.5.0/24 metric 10 up
      IP extended prefix: 10.222.29.0/24 metric 10 up
      IP extended prefix: 192.168.16.1/32 metric 0 up
    No queued transmissions
```

We see that Sherry is adjacent with three other IS-IS routers in the network: Sangiovese, Chianti, and Chablis. Each of these adjacencies prompts Sherry to advertise a TLV into the network that contains the neighbor ID of the peer, the IS-IS metric used to reach that neighbor, and the interface addresses for both the local and remote routers. Each TLV also includes traffic engineering information pertinent to the type of physical interface being used. For example, the adjacency with the Sangiovese router is formed over an OC-3c interface because the `Maximum bandwidth` is reported to be 155.52Mbps. This bandwidth is also reported for both the `Current reservable bandwidth` and the `Maximum reservable bandwidth`, which tells us that the interface is not oversubscribed and that no current reservations are in place. This particular interface, as well as the other two, doesn't currently belong to any administrative groups. We see this in the `Administrative groups: 0` display output. Finally, the `ge-0/2/0.0` interface connected to the Chianti router has the `te-metric` set to 30 in the router's configuration. This prompts the TLV representing this adjacency to include the Traffic Engineering Default Metric sub-TLV. This special metric value is seen in the router output as `Traffic engineering metric: 30`. The other two router interfaces do not include this sub-TLV and use the default IS-IS metric value of 10 for the traffic engineering calculations.

CSPF Algorithm Steps

Once the TED is populated by the IGPs, RSVP consults it when a new LSP is configured on the ingress router. The router uses any defined constraints supplied by the user when running the CSPF algorithm. These constraints include bandwidth reservations, administrative groups, and priority values. The algorithm creates a logical copy of the TED and performs the following steps to locate a path through the network meeting the constraints of the LSP:

1. When a bandwidth reservation is requested for the LSP, the algorithm removes all links from the database that don't have enough unreserved bandwidth available at the setup priority level of the LSP.

2. When the LSP requires a network path that includes links belonging to an administrative group, the algorithm removes all links that don't contain the requested group value.
3. When the LSP requires a network path that excludes links belonging to an administrative group, the algorithm removes all links that currently contain the requested group value.
4. The algorithm calculates the shortest path from the ingress to the egress router using the remaining information in the database. When an explicit route is requested by the user, separate shortest-path calculations are performed. The first is run from the ingress router to the first hop in the route. The next is calculated from the first hop of the route to the next hop in the route, if one is defined. This step-by-step process is performed for each node defined in the route until the egress router is encountered.
5. When multiple equal-cost paths exist from the ingress to egress router, the path whose last-hop interface address matches the egress router address is selected. This algorithm step is helpful when the egress router address is not a loopback address.
6. If multiple equal-cost paths still exist, the algorithm selects the path with the fewest number of physical hops.
7. In the event that multiple equal-cost paths continue to exist, the algorithm selects one of the paths based on the load-balancing configuration of the LSP: random, most-fill, or least-fill.
8. The algorithm generates an ERO containing the physical interface address of each node along the path from ingress to egress router. This ERO is used by the RSVP process to signal and establish the LSP.

NOTE

If multiple LSPs are committed at the same time, the CSPF algorithm calculates paths for the LSP with the highest setup priority, followed by LSPs with lower priority values. We discuss priority values in the section "LSP Priority and Preemption," later in the chapter.

Let's examine some of the user constraints used by CSPF in greater detail.

Administrative Groups

One aspect of traffic engineering is the ability to control what types of traffic use certain network links. One method for reaching this administrative goal is the use of *administrative groups*. The group values are used to control which LSPs are established across the specific network links. Once the LSPs are set up, you then control what types of traffic use those network connections.

Each network interface is configured with one or more group values, which are propagated via the IGPs and placed into the TED. These advertisements are encoded as individual bit positions using a bit vector system, as shown in Figure 8.2.

FIGURE 8.2 Administrative groups bit vector

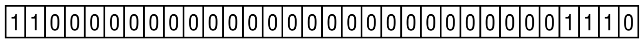
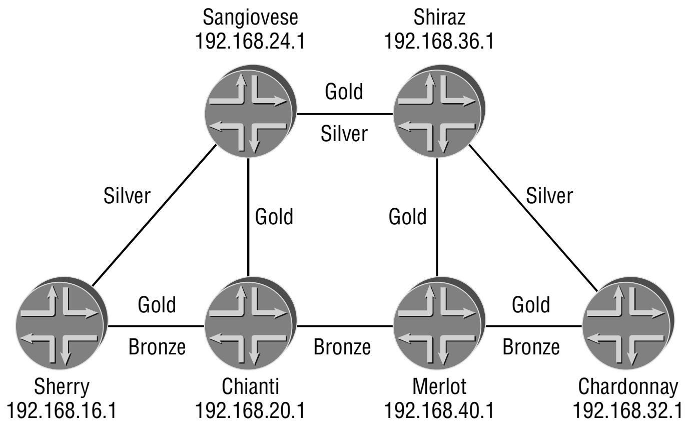


FIGURE 8.3 Administrative groups sample network



NOTE

A bit vector works in a fairly straightforward manner. Each group value is encoded as an individual bit. For example, a group value of 5 sets bit number 5 to a value of 1 in the vector. In a similar manner, a group value of 25 sets bit number 25 to a value of 1.

For ease of configuration and administration, the JUNOS software uses human-friendly names to locally represent the group values. These names are locally significant to the router because only the bit value is transmitted by the IGPs. Figure 8.3 shows a sample network with six routers running IS-IS and traffic engineering. The network administrators have defined certain links as Gold (a value of 2), others as Silver (a value of 15), and others as Bronze (a value of 28). The configuration of the Sherry router is currently set to

```
user@Sherry> show configuration protocols mpls
admin-groups {
    Gold 2;
    Silver 15;
    Bronze 28;
}
interface all;
interface at-0/1/0.0 {
    admin-group Silver;
}
interface ge-0/2/0.0 {
    admin-group [ Gold Bronze ];
}
```

This prompts Sherry to advertise the following traffic engineering information:

```
user@Sherry> **show isis database Sherry.00-00 extensive | find TLV**
  TLVs:
    Area address: 49.1111 (3)
    Speaks: IP
    Speaks: IPv6
    IP router id: 192.168.16.1
    IP address: 192.168.16.1
    Hostname: Sherry
    IS extended neighbor: Sangiovese.00, Metric: default 10
      IP address: 10.222.28.1
      Neighbor's IP address: 10.222.28.2
      Current reservable bandwidth:
        Priority 0 : 155.52Mbps
        Priority 1 : 155.52Mbps
        Priority 2 : 155.52Mbps
        Priority 3 : 155.52Mbps
        Priority 4 : 155.52Mbps
        Priority 5 : 155.52Mbps
        Priority 6 : 155.52Mbps
        Priority 7 : 155.52Mbps
      Maximum reservable bandwidth: 155.52Mbps
      Maximum bandwidth: 155.52Mbps
      Administrative groups:  0x8000  Silver
    IS extended neighbor: Chianti.02, Metric: default 10
      IP address: 10.222.29.1
      Traffic engineering metric: 30
      Current reservable bandwidth:
        Priority 0 : 1000Mbps
        Priority 1 : 1000Mbps
        Priority 2 : 1000Mbps
        Priority 3 : 1000Mbps
        Priority 4 : 1000Mbps
        Priority 5 : 1000Mbps
        Priority 6 : 1000Mbps
        Priority 7 : 1000Mbps
      Maximum reservable bandwidth: 1000Mbps
      Maximum bandwidth: 1000Mbps
      Administrative groups:  0x10000004  Bronze Gold
    IP extended prefix: 10.222.28.0/24 metric 10 up
    IP extended prefix: 10.222.29.0/24 metric 10 up
    IP extended prefix: 192.168.16.1/32 metric 0 up
  No queued transmissions
```

We examine the information in the TED directly through the `show ted database` command:

```
user@Sherry> show ted database Sherry.00 extensive
TED database: 8 ISIS nodes 6 INET nodes
NodeID: Sherry.00(192.168.16.1)
  Type: Rtr, Age: 284 secs, LinkIn: 2, LinkOut: 2
  Protocol: IS-IS(2)
    To: Sangiovese.00(192.168.24.1), Local: 10.222.28.1, Remote: 10.222.28.2
      Color: 0x8000 Silver
      Metric: 10
      Static BW: 155.52Mbps
      Reservable BW: 155.52Mbps
      Available BW [priority] bps:
       [0] 155.52Mbps   [1] 155.52Mbps   [2] 155.52Mbps   [3] 155.52Mbps
       [4] 155.52Mbps   [5] 155.52Mbps   [6] 155.52Mbps   [7] 155.52Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 155.52Mbps   [1] 155.52Mbps   [2] 155.52Mbps   [3] 155.52Mbps
         [4] 155.52Mbps   [5] 155.52Mbps   [6] 155.52Mbps   [7] 155.52Mbps
    To: Chianti.02, Local: 10.222.29.1, Remote: 0.0.0.0
      Color: 0x10000004 Bronze Gold
      Metric: 30
      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
```

Both outputs report that the interface connected to the Sangiovese router is set with the group name of `Silver`. This administrative group is assigned to the 15th bit in the group vector, which results in the hexadecimal output of 0x8000.

NOTE Remember that the least significant bit in the administrative group vector is bit position 0 and the most significant bit is position number 31.

The interface connecting Sherry to Chianti is currently assigned two administrative groups—Gold and Bronze. Both group values are present in the group vector by using the 0x10000004 notation to represent bit positions 28 and 2, respectively.

Once the network links have been assigned the appropriate groups values, we constrain the setup of an LSP by referencing the group names. The JUNOS software keyword of `include` means that the LSP may only use links that contain any one of the specified groups. For example, the ***Sherry-to-Char*** LSP is established from Sherry to Chardonnay using only links that belong to the Silver administrative group:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
admin-group {
    include Silver;
}
```

The LSP transits the Sangiovese and Shiraz routers before terminating on Chardonnay. We can examine the received Record Route object (RRO) for the LSP and verify that these addresses belong to the appropriate routers:

```
user@Sherry> show mpls lsp detail ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath:  (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Include: Silver
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.28.2 S 10.222.4.2 S 10.222.44.2 S
    Received RRO:
          10.222.28.2 10.222.4.2 10.222.44.2
Total 1 displayed, Up 1, Down 0

user@Sangiovese> show route 10.222.28.2/32 terse
```

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.28.2/32     L   0                        Local

user@Shiraz> show route 10.222.4.2/32 terse

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.4.2/32      L   0                        Local

user@Chardonnay> show route 10.222.44.2/32 terse

inet.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.44.2/32     L   0                        Local
```

In a similar manner, the ***Char-to-Sherry*** LSP is configured on the Chardonnay router to only use links belonging to the Gold administrative group. This LSP establishes across five separate links in the network:

```
[edit protocols mpls]
user@Chardonnay# show label-switched-path Char-to-Sherry
to 192.168.16.1;
admin-group {
    include Gold;
}

[edit protocols mpls]
user@Chardonnay# run show mpls lsp detail ingress
Ingress LSP: 1 sessions

192.168.16.1
  From: 192.168.32.1, State: Up, ActiveRoute: 0, LSPname: Char-to-Sherry
  ActivePath:  (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
```

```
    Include: Gold
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 50)
          10.222.45.1 S 10.222.46.2 S 10.222.4.1 S 10.222.30.2 S 10.222.29.1 S
    Received RRO:
          10.222.45.1 10.222.46.2 10.222.4.1 10.222.30.2 10.222.29.1
Total 1 displayed, Up 1, Down 0
```

An interesting problem occurs when we add the JUNOS keyword `exclude` to the ***Sherry-to-Char*** LSP on the Sherry router. The `exclude` command requires that the LSP not use any links that contain the specified group value. In this particular case, we inform the LSP to avoid the Gold links in the network. The configuration is updated to:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
admin-group {
    include Silver;
    exclude Gold;
}
```

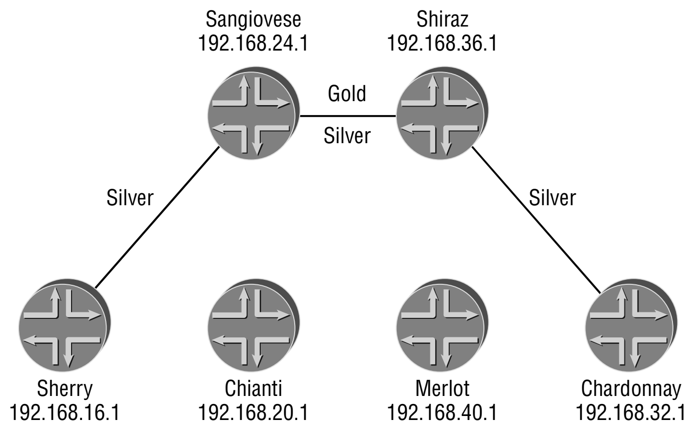
A quick look at the output of the `show mpls lsp ingress` command informs us that the LSP is no longer operational:

```
user@Sherry> show mpls lsp ingress
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Dn     0 -                      Sherry-to-Char
Total 1 displayed, Up 0, Down 1
```

When we add the *extensive* option to the command, we can see the history log for the LSP:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Dn, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: (none)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary                   State: Dn
    Include: Silver    Exclude: Gold
    Will be enqueued for recomputation in 9 second(s).
    1 Jun  4 16:24:53  CSPF failed: no route toward 192.168.32.1[6 times]
  Created: Wed Jun  4 16:19:41 2003
Total 1 displayed, Up 0, Down 1
```

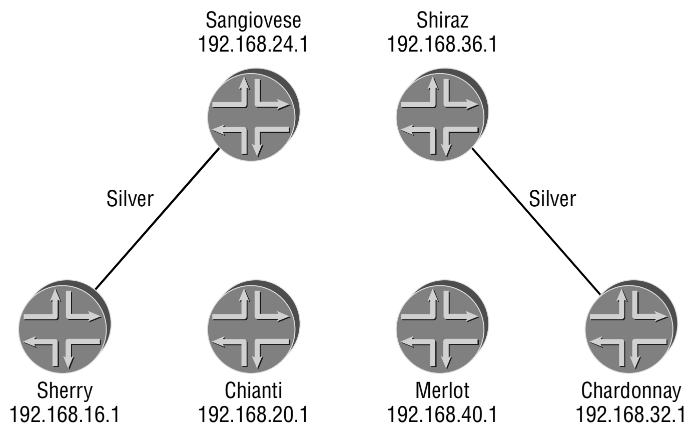
FIGURE 8.4 CSPF network pruning for include groups



The `CSPF failed: no route toward 192.168.32.1` output simply tells us that CSPF can't find a route from the ingress to egress routers, which also meets the constraints we provided. In this particular case we know that we've only asked for administrative group constraints, so we can consult the TED and examine the information each router is advertising to see why the LSP didn't set up. Instead of doing that here, let's take a graphical look at how CSPF operates.

Since we haven't requested any bandwidth reservations for the LSP, the algorithm first removes all network links that do not include the administrative group of Silver. The result of this database pruning is seen in Figure 8.4.

FIGURE 8.5 CSPF network pruning for exclude groups



The algorithm then removes all links from the database that currently contain the excluded administrative group of Gold. The remaining links that meet the constraints are seen in Figure 8.5. At this point, CSPF then attempts to find a shortest path from the ingress to the egress router. Clearly, there is no connectivity between these two routers using the constraints, so the LSP is not established in the network.

LSP Priority and Preemption

During our discussion of the steps performed by the CSPF algorithm, we referred to the priority of the LSP. In reality, each LSP has two distinct values it uses: a setup and a hold priority. The *setup priority* value is used to determine if enough bandwidth is available at that priority level (between 0 and 7) to establish the LSP. The *hold priority* value is used by an established LSP to retain its bandwidth reservations in the network. As a backdrop for discussing these priority values, let's examine the information injected into the TED by the Sherry router in Figure 8.3:

```
user@Sherry> show ted database Sherry.00 extensive | find Chianti
    To: Chianti.02, Local: 10.222.29.1, Remote: 0.0.0.0
      Color: 0 <none>
      Metric: 10
      Static BW: 1000Mbps
      Reservable BW: 1000Mbps
      Available BW [priority] bps:
       [0] 1000Mbps     [1] 1000Mbps     [2] 1000Mbps     [3] 1000Mbps
       [4] 1000Mbps     [5] 1000Mbps     [6] 1000Mbps     [7] 1000Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 1000Mbps     [1] 1000Mbps     [2] 1000Mbps     [3] 1000Mbps
         [4] 1000Mbps     [5] 1000Mbps     [6] 1000Mbps     [7] 1000Mbps
```

Each of the router interfaces is reporting available bandwidth values at each of the eight priority levels, with priority level 0 being the best and level 7 the worst. When you configure an LSP for a bandwidth reservation, the CSPF algorithm consults the bandwidth availability at the priority level corresponding to the LSP's setup priority value. By default, all LSPs created by the JUNOS software receive a setup priority value of 7, so the bandwidth at the lowest level is consulted. This default setting ensures that a newly establishing LSP doesn't consume bandwidth from an established LSP. Once CSPF finds a path through the network and creates the strict-hop ERO, the RSVP signaling process establishes the path. As the `Resv` message travels upstream to the ingress, the actual bandwidth on each interface is reserved at the LSP's hold priority value, which is 0 by default, as well as all of the lower priority values. This ensures that an established LSP can't have its bandwidth reservations taken away by a new LSP in the network. Once the bandwidth reservation has been accounted for on each router, updated TED information is generated and propagated throughout the network.

By default, the JUNOS software advertises TED information after a change of 10 percent of the interface's available bandwidth. You can alter this percentage between 1 and 20 percent with the `update-threshold` command. This is applied to individual interfaces within the `[edit protocols rsvp]` configuration hierarchy.

Using Figure 8.3 as a guide, we configure an LSP on the Sherry router to reserve 100Mbps of bandwidth along the network path through Chianti. The configuration of the LSP looks like this:

```
user@Sherry> show configuration protocols mpls
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    bandwidth 100m;
    primary via-Chianti;
}
path via-Chianti {
    192.168.20.1 loose;
}
interface all;
```

After verifying that the LSP is operational, we examine the TED information advertised by Sherry to the network:

```
user@Sherry> show mpls lsp ingress
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0 via-Chianti      *     Sherry-to-Char
Total 1 displayed, Up 1, Down 0

user@Sherry> show ted database Sherry.00 extensive | find Chianti
    To: Chianti.02, Local: 10.222.29.1, Remote: 0.0.0.0
      Color: 0 <none>
      Metric: 10
      Static BW: 1000Mbps
      Reservable BW: 1000Mbps
      Available BW [priority] bps:
       [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
       [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
         [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
```

LSP Priority Values

When you have a requirement to alter the default setup and hold priority values, it is a recommended good practice to configure them to the same value. This makes troubleshooting easier for operations personnel and provides an easy method for evaluating which LSP is more important than a second LSP. For example, an LSP with a priority setting of `1 1` will clearly preempt an LSP with a priority setting of `5 5`.

Technically speaking, the priority values may be different, but the setup value must always be equal to or less than the hold priority. To illustrate why this restriction is in place, suppose that both LSP-1 and LSP-2 have their priority values set to `2 5`. This means that the setup priority is 2 while the hold priority is 5. When LSP-1 is established in the network, it examines the bandwidth reservations at priority level 2 but only places a reservation at level 5 and below. The ingress router now attempts to establish LSP-2 and examines the bandwidth reservations at priority level 2. Since enough bandwidth is available in the network, the LSP is established and preempts LSP-1 from the network. However, the bandwidth reservation for LSP-2 is also only made at level 5 and below. LSP-1 now reattempts to establish itself and sees available bandwidth at priority level 2 and preempts LSP-2 from the network. This vicious cycle continues repeating itself, with one LSP preempting the other over and over again. To prevent this situation from ever occurring, the setup priority must be equal to or less than the hold priority.

The TED information reports that 100Mbps of bandwidth was reserved at all eight priority levels along the link between Sherry and Chianti. This correlates to the default setup and hold values of 7 and 0, respectively.

The JUNOS software provides you with the ability to designate some LSPs as being more important than others by manually configuring the priority values via the `priority` command. This means that you could have a new LSP *preempt* an established LSP from the network when there is contention for bandwidth resources. For example, suppose that all of the network links are Fast Ethernet connections and that an LSP is established with a bandwidth reservation of 90Mbps. When the default priority values are used, any new LSP requesting a bandwidth value greater than 10Mbps won't be established in the network. Now suppose that the 90Mbps established LSP has a hold priority of 3. Any new LSP with a setup priority value of 2 or better, regardless of its requested bandwidth amount, will be able to set up in the network. In this situation, the 90Mbps LSP is said to be preempted from the network by a higher priority LSP. See the sidebar "LSP Priority Values" to learn how this works in further detail.

The Sherry router now configures an LSP called ***`low-pri-LSP`*** along the network path through Sangiovese. This LSP is requesting a bandwidth reservation of 80Mbps and has its setup and hold priority values both set to 3. Its configuration looks like this:

```
user@Sherry> show configuration protocols mpls
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    bandwidth 100m;
    primary via-Chianti;
}
label-switched-path low-pri-LSP {
    to 192.168.32.1;
    bandwidth 80m;
    priority 3 3;
    primary via-Sangiovese;
}
path via-Chianti {
    192.168.20.1 loose;
}
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;
```

In addition to establishing the LSP, the 80Mbps of bandwidth reservation is removed from the links along the LSP's path. Specifically, the link between Sherry and Sangiovese now reports only 75.52Mbps of available bandwidth at the lower priority levels:

```
user@Sherry> show mpls lsp ingress name low-pri-LSP
Ingress LSP: 2 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0 via-Sangiovese   *     low-pri-LSP
Total 1 displayed, Up 1, Down 0

user@Sherry> show ted database Sherry.00 extensive
TED database: 11 ISIS nodes 9 INET nodes
NodeID: Sherry.00(192.168.16.1)
  Type: Rtr, Age: 225 secs, LinkIn: 2, LinkOut: 2
  Protocol: IS-IS(2)
    To: Sangiovese.00(192.168.24.1), Local: 10.222.28.1, Remote: 10.222.28.2
      Color: 0 <none>
      Metric: 10
      Static BW: 155.52Mbps
      Reservable BW: 155.52Mbps
      Available BW [priority] bps:
       [0] 155.52Mbps   [1] 155.52Mbps   [2] 155.52Mbps   [3] 75.52Mbps
       [4] 75.52Mbps    [5] 75.52Mbps    [6] 75.52Mbps    [7] 75.52Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 155.52Mbps   [1] 155.52Mbps   [2] 155.52Mbps   [3] 75.52Mbps
         [4] 75.52Mbps    [5] 75.52Mbps    [6] 75.52Mbps    [7] 75.52Mbps
    To: Chianti.02, Local: 10.222.29.1, Remote: 0.0.0.0
      Color: 0 <none>
      Metric: 10
      Static BW: 1000Mbps
      Reservable BW: 1000Mbps
      Available BW [priority] bps:
       [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
       [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
         [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
```

A new LSP, called ***high-pri-LSP***, is now created on the Sherry router. While it is requesting 90Mbps of bandwidth, it is also using the default priority values:

```
user@Sherry> show configuration protocols mpls
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    bandwidth 100m;
    primary via-Chianti;
}
label-switched-path low-pri-LSP {
    to 192.168.32.1;
    bandwidth 80m;
    priority 3 3;
    primary via-Sangiovese;
}
label-switched-path high-pri-LSP {
    to 192.168.32.1;
    bandwidth 90m;
    primary via-Sangiovese;
}
path via-Chianti {
    192.168.20.1 loose;
}
```

```
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;
```

Because the new LSP is using a setup priority value of 7, only the available bandwidth at that level is evaluated in the TED. This means that not enough capacity is available for the new LSP to establish itself and that it is currently in a Down state:

```
user@Sherry> show mpls lsp ingress name high-pri-LSP
Ingress LSP: 3 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    0.0.0.0         Dn     0 -                      high-pri-LSP
Total 1 displayed, Up 0, Down 1
```

The network administrators have decided that the ***high-pri-LSP*** LSP should have better access to the network's resources, so they have assigned it a setup and hold priority value of 1:

```
user@Sherry> show configuration protocols mpls | find high-pri-LSP
label-switched-path high-pri-LSP {
    to 192.168.32.1;
    bandwidth 90m;
    priority 1 1;
    primary via-Sangiovese;
}
path via-Chianti {
    192.168.20.1 loose;
}
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;
```

The LSP is now fully established in the network and the Sherry router has altered its TED information to reflect the new bandwidth reservation:

```
user@Sherry> show mpls lsp ingress name high-pri-LSP
Ingress LSP: 3 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0 via-Sangiovese   *     high-pri-LSP
Total 1 displayed, Up 1, Down 0

user@Sherry> show ted database Sherry.00 extensive
TED database: 11 ISIS nodes 9 INET nodes
```

```
NodeID: Sherry.00(192.168.16.1)
  Type: Rtr, Age: 108 secs, LinkIn: 2, LinkOut: 2
  Protocol: IS-IS(2)
    To: Sangiovese.00(192.168.24.1), Local: 10.222.28.1, Remote: 10.222.28.2
      Color: 0 <none>
      Metric: 10
      Static BW: 155.52Mbps
      Reservable BW: 155.52Mbps
      Available BW [priority] bps:
       [0] 155.52Mbps   [1] 65.52Mbps    [2] 65.52Mbps    [3] 65.52Mbps
       [4] 65.52Mbps    [5] 65.52Mbps    [6] 65.52Mbps    [7] 65.52Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 155.52Mbps   [1] 65.52Mbps    [2] 65.52Mbps    [3] 65.52Mbps
         [4] 65.52Mbps    [5] 65.52Mbps    [6] 65.52Mbps    [7] 65.52Mbps
    To: Chianti.02, Local: 10.222.29.1, Remote: 0.0.0.0
      Color: 0 <none>
      Metric: 10
      Static BW: 1000Mbps
      Reservable BW: 1000Mbps
      Available BW [priority] bps:
       [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
       [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
      Interface Switching Capability Descriptor(1):
        Switching type: Packet
        Encoding type: Packet
        Maximum LSP BW [priority] bps:
         [0] 900Mbps      [1] 900Mbps      [2] 900Mbps      [3] 900Mbps
         [4] 900Mbps      [5] 900Mbps      [6] 900Mbps      [7] 900Mbps
```

In addition to seeing that the ***low-pri-LSP*** is in a `Down` state, we can locate further information by examining the output of the `show mpls lsp extensive` command. Within the LSP history portion, we see that the LSP was preempted from the network with the `Session preempted` output:

```
user@Sherry> show mpls lsp ingress
Ingress LSP: 3 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Dn     0 -                      low-pri-LSP
192.168.32.1    192.168.16.1    Up     0 via-Sangiovese   *     high-pri-LSP
192.168.32.1    192.168.16.1    Up     0 via-Chianti      *     Sherry-to-Char
```

```
Total 3 displayed, Up 2, Down 1

user@Sherry> show mpls lsp extensive ingress name low-pri-LSP
Ingress LSP: 3 sessions

192.168.32.1
  From: 192.168.16.1, State: Dn, ActiveRoute: 0, LSPname: low-pri-LSP
  ActivePath: (none)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary   via-Sangiovese   State: Dn
    Priorities: 3 3
    Bandwidth: 80Mbps
    Will be enqueued for recomputation in 28 second(s).
   15 Jun  4 23:48:20  CSPF failed: no route toward 192.168.32.1[21 times]
   14 Jun  4 23:38:46  Clear Call
   13 Jun  4 23:38:46  Deselected as active
   12 Jun  4 23:38:46  10.222.28.2: Requested bandwidth unavailable
   11 Jun  4 23:38:46  ResvTear received
   10 Jun  4 23:38:46  Down
    9 Jun  4 23:38:46  Change in active path
    8 Jun  4 23:38:46  10.222.4.2: Requested bandwidth unavailable
    7 Jun  4 23:38:46  CSPF failed: no route toward 192.168.32.1
    6 Jun  4 23:38:46  10.222.4.2: Session preempted
    5 Jun  4 23:25:29  Selected as active path
    4 Jun  4 23:25:29  Record Route:  10.222.28.2 10.222.4.2 10.222.44.2
    3 Jun  4 23:25:29  Up
    2 Jun  4 23:25:29  Originate Call
    1 Jun  4 23:25:29  CSPF: computation result accepted
  Created: Wed Jun  4 23:22:27 2003
Total 1 displayed, Up 0, Down 1
```

LSP Traffic Protection

When an established LSP experiences a failure and is torn down, the traffic that was transiting the LSP begins to be forwarded using native IPv4 lookups in the network. While this may not sound catastrophic, many network administrators prefer to always use an LSP for forwarding traffic. The reasons for this desire range from collecting accurate statistics for billing to guaranteeing levels of service to meet contractual arrangements. Within the JUNOS software, there are three methods for protecting traffic in the network. These include the use of primary paths, secondary paths, and fast reroute. Let's examine each in some detail.

Primary LSP Paths

As this point in your JUNOS software education, you're already aware of primary paths and have configured them on an LSP. After all, this is the method for assigning a user-defined ERO to an LSP. However, much more goes along with a primary path than this. This path definition may contain multiple user constraints, such as bandwidth reservations, priority values, user-defined EROs, and administrative groups. When the CSPF algorithm evaluates the TED, it uses the constraints provided within the primary path to locate a usable path.

An individual LSP can have zero or one *primary path* applied. When it is configured, the path must be used if it's available in the network. This means that the primary path carries with it a *revertive* capability. To better explain what this means, assume that an LSP is established in the network using a primary path. At some point, one of the path's links fails and the LSP is moved to a backup network path. When the primary path once again becomes usable, the LSP will reestablish along that path.

The revertive capability of the primary path is controlled by two configuration variables. The `retry-timer` controls the length of time that the ingress router waits between attempts to reestablish the primary path, and the `retry-limit` controls how many times the ingress attempts this reestablishment. The default value for the `retry-timer` is 30 seconds and can be set between 1 and 600 seconds (10 minutes). Once the primary path is reestablished, the ingress router waits for two instances of the timer to expire before actually using the primary path. This prevents potential thrashing in the network by ensuring that the new path is stable for a period of time before its use. By default, the JUNOS software sets the `retry-limit` to a value of 0, which means that the ingress router continually attempts to reestablish the primary path. You can set the limit to as high as 10,000, at which point the ingress router stops attempting to reestablish the path. If a viable network path becomes available after the limit is reached, you must manually clear the LSP on the ingress router to once again use the primary path.

FIGURE 8.6 Primary/secondary paths sample network

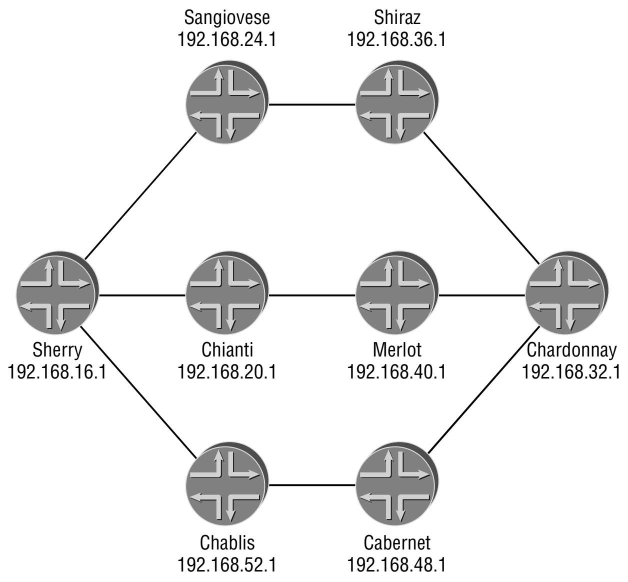


Figure 8.6 shows a network consisting of eight routers in an MPLS network. The ***Sherry-to-Char*** LSP is configured on the Sherry router with a primary path defined that reserves 20Mbps of bandwidth. The path definition also applies a user-defined ERO called ***via-Chablis***, which lists the loopback address of the Chablis router as a loose hop. Here is the configuration of the LSP:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
primary via-Chablis {
    bandwidth 20m;
}
```

The LSP is operational and is using Chablis as a transit router:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
    5 Jun  6 14:35:50  Selected as active path
    4 Jun  6 14:35:50  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:35:50  Up
    2 Jun  6 14:35:50  Originate Call
    1 Jun  6 14:35:50  CSPF: computation result accepted
  Created: Fri Jun  6 14:34:19 2003
Total 1 displayed, Up 1, Down 0
```

Secondary LSP Paths

An individual LSP may define one or more secondary paths to provide for the continual forwarding of traffic should the primary path experience a failure. Each configured *secondary path* may contain multiple user constraints for use by the CSPF algorithm, and each path is treated in an equal manner by the LSP. This means that the path definitions don't carry with them any inherent priority for being established first. When the ingress router is notified of a failure along the primary path, it attempts to establish the first secondary path it locates in the configuration. Once this secondary path is established, user traffic again uses the LSP for forwarding across the network.

The ***Sherry-to-Char*** LSP is now configured with a secondary path named ***via-Chianti***. This path definition requests a bandwidth reservation of 10Mbps from the network and assigns an ERO, which prompts the path to use Chianti as a transit router:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
primary via-Chablis {
    bandwidth 20m;
}
secondary via-Chianti {
    bandwidth 10m;
}
```

Once the configuration is committed, the secondary path is known to the LSP. We can see this information in the output of the `show mpls lsp extensive ingress` command, where the secondary path appears after the primary path:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
    5 Jun  6 14:51:54  Selected as active path
    4 Jun  6 14:51:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:51:54  Up
    2 Jun  6 14:51:54  Originate Call
    1 Jun  6 14:51:54  CSPF: computation result accepted
  Secondary via-Chianti      State: Dn
    Bandwidth: 10Mbps
        No computed ERO.
  Created: Fri Jun  6 14:50:03 2003
Total 1 displayed, Up 1, Down 0
```

The secondary path is currently not operational, as reported by the `State: Dn` output. This is the normal operating state for a secondary path when the primary path is functioning. The `ActivePath:` portion of the router output provides a clue as to which path the LSP is using to forward traffic across the network. In a similar manner to the routing table, the presence of the asterisk (*) next to the path definition tells us that the primary path is the active path.

When the network link between Chianti and Cabernet in Figure 8.6 fails, the primary path of the LSP is torn down. This prompts the ingress router to establish the secondary path through the Chianti router:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary   via-Chablis      State: Dn
    Bandwidth: 20Mbps
    Will be enqueued for recomputation in 6 second(s).
   14 Jun  6 14:59:40  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 14:59:37  Clear Call
   12 Jun  6 14:59:37  CSPF: link down/deleted 10.222.60.1(Chablis.00/
  192.168.52.1)->10.222.60.2(Cabernet.00/192.168.48.1)
   11 Jun  6 14:59:37  Deselected as active
   10 Jun  6 14:59:37  ResvTear received
    9 Jun  6 14:59:37  Down
    8 Jun  6 14:59:37  Change in active path
    7 Jun  6 14:59:37  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 14:59:37  10.222.5.2: No Route toward dest
    5 Jun  6 14:51:54  Selected as active path
    4 Jun  6 14:51:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:51:54  Up
    2 Jun  6 14:51:54  Originate Call
    1 Jun  6 14:51:54  CSPF: computation result accepted
 *Secondary via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 14:59:37  Selected as active path
    4 Jun  6 14:59:37  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 14:59:37  Up
    2 Jun  6 14:59:37  Originate Call
    1 Jun  6 14:59:37  CSPF: computation result accepted
  Created: Fri Jun  6 14:51:44 2003
Total 1 displayed, Up 1, Down 0
```

NOTE

The output of the `CSPF: link down/deleted` history entry exceeds the 80-character limit of most terminal windows. It is included here in its entirety for completeness. Future router output will truncate the entry for readability.

The router output tells us that the primary path is currently down and the secondary path is in an `Up` state. When the ingress router received the `ResvTear` message from its neighbor, it cleared the primary path from the network. It then ran the CSPF algorithm using the constraints provided by the secondary path and found a usable set of links that met the constraints. The resulting ERO from the CSPF calculation was passed to the RSVP process and the secondary path was signaled. Once it became established, the router once again used it for forwarding user traffic across the network.

After clearing the primary path, the ingress router begins the 30-second retry timer for the path. When it expires, the CSPF algorithm is run in an attempt to locate a new path for the primary to use. After two unsuccessful attempts, the Sherry router finds the Chablis-Cabernet link restored and the primary path is re-signaled:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
   18 Jun  6 15:00:09  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   17 Jun  6 15:00:09  Up
   16 Jun  6 15:00:09  Originate Call
   15 Jun  6 15:00:09  CSPF: computation result accepted
   14 Jun  6 14:59:40  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 14:59:37  Clear Call
   12 Jun  6 14:59:37  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   11 Jun  6 14:59:37  Deselected as active
   10 Jun  6 14:59:37  ResvTear received
    9 Jun  6 14:59:37  Down
    8 Jun  6 14:59:37  Change in active path
    7 Jun  6 14:59:37  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 14:59:37  10.222.5.2: No Route toward dest
    5 Jun  6 14:51:54  Selected as active path
    4 Jun  6 14:51:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:51:54  Up
    2 Jun  6 14:51:54  Originate Call
    1 Jun  6 14:51:54  CSPF: computation result accepted
 *Secondary via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 14:59:37  Selected as active path
    4 Jun  6 14:59:37  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 14:59:37  Up
    2 Jun  6 14:59:37  Originate Call
    1 Jun  6 14:59:37  CSPF: computation result accepted
  Created: Fri Jun  6 14:51:34 2003
Total 1 displayed, Up 1, Down 0
```

Even though the primary path is now in an Up state, the secondary path is still the active path for the LSP. This is due to the waiting period imposed by the JUNOS software before using a newly established primary path. This period is defined as two iterations of the `retry-timer` setting, set to 30 seconds in our case. In reality, you might not always see the router wait for exactly 60 seconds. This has to do with the randomization of the timers within the router as well as when the LSP references the timer value. In this particular example, it appears as if the ingress router waited 29 seconds before selecting the primary path as the active path for the LSP:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
   19 Jun  6 15:00:38  Selected as active path
   18 Jun  6 15:00:09  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   17 Jun  6 15:00:09  Up
   16 Jun  6 15:00:09  Originate Call
   15 Jun  6 15:00:09  CSPF: computation result accepted
   14 Jun  6 14:59:40  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 14:59:37  Clear Call
   12 Jun  6 14:59:37  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   11 Jun  6 14:59:37  Deselected as active
   10 Jun  6 14:59:37  ResvTear received
    9 Jun  6 14:59:37  Down
    8 Jun  6 14:59:37  Change in active path
    7 Jun  6 14:59:37  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 14:59:37  10.222.5.2: No Route toward dest
    5 Jun  6 14:51:54  Selected as active path
    4 Jun  6 14:51:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:51:54  Up
    2 Jun  6 14:51:54  Originate Call
    1 Jun  6 14:51:54  CSPF: computation result accepted
  Secondary via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    6 Jun  6 15:00:38  Deselected as active
    5 Jun  6 14:59:37  Selected as active path
    4 Jun  6 14:59:37  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 14:59:37  Up
    2 Jun  6 14:59:37  Originate Call
    1 Jun  6 14:59:37  CSPF: computation result accepted
  Created: Fri Jun  6 14:51:08 2003
Total 1 displayed, Up 1, Down 0
```

It is interesting to note that the secondary path is still in an Up state. In fact, the router uses the same waiting period for clearing the secondary path as it used for activating the primary path. This helps to ensure a quicker recovery time in case the primary path rapidly fails after being selected as active. In our example, it appears as if the router waits for 1 minute and 29 seconds before removing the secondary path from the network:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
   19 Jun  6 15:00:38  Selected as active path
   18 Jun  6 15:00:09  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   17 Jun  6 15:00:09  Up
   16 Jun  6 15:00:09  Originate Call
   15 Jun  6 15:00:09  CSPF: computation result accepted
   14 Jun  6 14:59:40  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 14:59:37  Clear Call
   12 Jun  6 14:59:37  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   11 Jun  6 14:59:37  Deselected as active
   10 Jun  6 14:59:37  ResvTear received
    9 Jun  6 14:59:37  Down
    8 Jun  6 14:59:37  Change in active path
    7 Jun  6 14:59:37  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 14:59:37  10.222.5.2: No Route toward dest
    5 Jun  6 14:51:54  Selected as active path
    4 Jun  6 14:51:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 14:51:54  Up
    2 Jun  6 14:51:54  Originate Call
    1 Jun  6 14:51:54  CSPF: computation result accepted
  Secondary via-Chianti      State: Dn
    Bandwidth: 10Mbps
```

```
        No computed ERO.
    7 Jun  6 15:02:07  Clear Call
    6 Jun  6 15:00:38  Deselected as active
    5 Jun  6 14:59:37  Selected as active path
    4 Jun  6 14:59:37  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 14:59:37  Up
    2 Jun  6 14:59:37  Originate Call
    1 Jun  6 14:59:37  CSPF: computation result accepted
  Created: Fri Jun  6 14:49:21 2003
Total 1 displayed, Up 1, Down 0
```

Unfortunately, the granularity of the timestamps in the command output makes it appear as if the primary path was torn down and the secondary path was established instantaneously. In fact, this is not the case, and for some period of time, the ingress router was forwarding traffic using the IPv4 next hops in its routing table. This delay in the setup of the secondary path can be eliminated by establishing the secondary path before the primary path fails. Let's see how this works.

Standby Secondary Paths

Any or all of the secondary paths defined for an LSP can be signaled and established in the network once the primary path is operational. This is accomplished through the use of the `standby` command within the path definition. When applied, the ingress router waits for the primary path to fully become established in the network and be selected as the active path. It then consults the TED to locate a path that meets the constraints of the secondary path. If it locates a viable set of links, the ERO is passed to RSVP and the path is established and placed into the `Up` state.

The Sherry router in Figure 8.6 adds the `standby` option to the secondary path of ***via-Chianti***:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
primary via-Chablis {
    bandwidth 20m;
}
secondary via-Chianti {
    bandwidth 10m;
    standby;
}
```

This prompts the Sherry router to establish both the primary and secondary paths in the network:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
```

```
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
    6 Jun  6 15:41:25  Selected as active path
    5 Jun  6 15:41:25  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    4 Jun  6 15:41:25  Up
    3 Jun  6 15:41:25  Originate Call
    2 Jun  6 15:41:25  CSPF: computation result accepted
    1 Jun  6 15:40:55  CSPF failed: no route toward 192.168.52.1
  Standby   via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    4 Jun  6 15:41:25  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 15:41:25  Up
    2 Jun  6 15:41:25  Originate Call
    1 Jun  6 15:41:25  CSPF: computation result accepted
  Created: Fri Jun  6 15:39:25 2003
Total 1 displayed, Up 1, Down 0
```

NOTE The history buffers were cleared for ease of readability. This was accomplished by deactivating and then reactivating the MPLS configuration on the Sherry router.

Once again the Chablis-Cabernet network link in Figure 8.6 fails and causes the primary path of the LSP to be torn down. Because the secondary path is already in an Up state, the ingress router simply installs it as the active path for the LSP:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
```

```
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Primary   via-Chablis      State: Dn
    Bandwidth: 20Mbps
    Will be enqueued for recomputation in 18 second(s).
   14 Jun  6 15:46:17  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 15:46:17  Clear Call
   12 Jun  6 15:46:17  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   11 Jun  6 15:46:17  ResvTear received
   10 Jun  6 15:46:17  Down
    9 Jun  6 15:46:17  Deselected as active
    8 Jun  6 15:46:17  CSPF failed: no route toward 192.168.32.1
    7 Jun  6 15:46:17  10.222.5.2: No Route toward dest
    6 Jun  6 15:41:25  Selected as active path
    5 Jun  6 15:41:25  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    4 Jun  6 15:41:25  Up
    3 Jun  6 15:41:25  Originate Call
    2 Jun  6 15:41:25  CSPF: computation result accepted
    1 Jun  6 15:40:55  CSPF failed: no route toward 192.168.52.1
 *Standby   via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 15:46:17  Selected as active path
    4 Jun  6 15:41:25  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 15:41:25  Up
    2 Jun  6 15:41:25  Originate Call
    1 Jun  6 15:41:25  CSPF: computation result accepted
  Created: Fri Jun  6 15:40:45 2003
Total 1 displayed, Up 1, Down 0
```

After the link returns to service, the primary path is signaled and established but is not selected as the active path for the LSP. After waiting for the primary path to remain stable, the ingress router once again selects it as the active path. The secondary path is deselected but remains operational due to its `standby` configuration:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions
```

```
192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Bandwidth: 20Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
   19 Jun  6 15:47:44  Selected as active path
   18 Jun  6 15:46:46  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   17 Jun  6 15:46:46  Up
   16 Jun  6 15:46:46  Originate Call
   15 Jun  6 15:46:46  CSPF: computation result accepted
   14 Jun  6 15:46:17  CSPF failed: no route toward 192.168.32.1[2 times]
   13 Jun  6 15:46:17  Clear Call
   12 Jun  6 15:46:17  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   11 Jun  6 15:46:17  ResvTear received
   10 Jun  6 15:46:17  Down
    9 Jun  6 15:46:17  Deselected as active
    8 Jun  6 15:46:17  CSPF failed: no route toward 192.168.32.1
    7 Jun  6 15:46:17  10.222.5.2: No Route toward dest
    6 Jun  6 15:41:25  Selected as active path
    5 Jun  6 15:41:25  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    4 Jun  6 15:41:25  Up
    3 Jun  6 15:41:25  Originate Call
    2 Jun  6 15:41:25  CSPF: computation result accepted
    1 Jun  6 15:40:55  CSPF failed: no route toward 192.168.52.1
  Standby   via-Chianti      State: Up
    Bandwidth: 10Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    6 Jun  6 15:47:44  Deselected as active
    5 Jun  6 15:46:17  Selected as active path
    4 Jun  6 15:41:25  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 15:41:25  Up
    2 Jun  6 15:41:25  Originate Call
    1 Jun  6 15:41:25  CSPF: computation result accepted
  Created: Fri Jun  6 15:38:24 2003
Total 1 displayed, Up 1, Down 0
```

Using Only Secondary Paths

The use of a standby secondary path in conjunction with a primary path eases the failover of traffic between the paths. The revertive nature of the primary path, however, means that the ingress will move traffic back to the primary after it is available again. Some network administrators do not wish to incur the "extra" network churn that results from the revert action. They would prefer that the failure of the primary path cause traffic to use the secondary path and remain there indefinitely. While there is no mechanism to keep a primary path from reverting to active status, the JUNOS software allows you to configure multiple secondary paths on an LSP without defining a primary path. The ingress router uses these multiple secondary paths in the order they appear in the configuration.

Within the network displayed in Figure 8.6, the ***Sherry-to-Char*** LSP is reconfigured to contain two secondary paths—***via-Chablis*** and ***via-Chianti***:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
secondary via-Chablis;
secondary via-Chianti;
```

The Sherry router runs the CSPF algorithm against the contents of the TED for the first secondary path listed, ***via-Chablis***, in an attempt to locate a viable set of links that meet the constraints. The output of the `show mpls lsp extensive` command tells us that this path setup was successful:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Secondary via-Chablis      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
    5 Jun  6 16:45:30  Selected as active path
    4 Jun  6 16:45:30  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 16:45:30  Up
    2 Jun  6 16:45:30  Originate Call
    1 Jun  6 16:45:30  CSPF: computation result accepted
  Secondary via-Chianti      State: Dn
        No computed ERO.
  Created: Fri Jun  6 16:43:54 2003
Total 1 displayed, Up 1, Down 0
```

When the Chablis-Cabernet link once again fails, the ***via-Chianti*** secondary path is signaled and established in the network. This then becomes the active path for the LSP:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Secondary via-Chablis      State: Dn
    Will be enqueued for recomputation in 6 second(s).
   13 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
   12 Jun  6 16:49:39  Clear Call
   11 Jun  6 16:49:39  CSPF: link down/deleted 10.222.60.1(Chablis.00
   10 Jun  6 16:49:39  Deselected as active
    9 Jun  6 16:49:39  ResvTear received
    8 Jun  6 16:49:39  Down
    7 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 16:49:39  10.222.5.2: No Route toward dest
    5 Jun  6 16:45:30  Selected as active path
    4 Jun  6 16:45:30  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 16:45:30  Up
    2 Jun  6 16:45:30  Originate Call
    1 Jun  6 16:45:30  CSPF: computation result accepted
 *Secondary via-Chianti      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 16:49:39  Selected as active path
    4 Jun  6 16:49:39  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 16:49:39  Up
    2 Jun  6 16:49:39  Originate Call
    1 Jun  6 16:49:39  CSPF: computation result accepted
  Created: Fri Jun  6 16:45:22 2003
Total 1 displayed, Up 1, Down 0
```

After the link returns to service, the ingress router reestablishes the ***via-Chablis*** path since it was previously the active path for the LSP. The current active path for the LSP doesn't change, however, and remains ***via-Chianti***:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Secondary via-Chablis      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2 10.222.60.2 10.222.6.2
   17 Jun  6 16:49:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   16 Jun  6 16:49:54  Up
   15 Jun  6 16:49:54  Originate Call
   14 Jun  6 16:49:54  CSPF: computation result accepted
   13 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
   12 Jun  6 16:49:39  Clear Call
   11 Jun  6 16:49:39  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   10 Jun  6 16:49:39  Deselected as active
    9 Jun  6 16:49:39  ResvTear received
    8 Jun  6 16:49:39  Down
    7 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 16:49:39  10.222.5.2: No Route toward dest
    5 Jun  6 16:45:30  Selected as active path
    4 Jun  6 16:45:30  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 16:45:30  Up
    2 Jun  6 16:45:30  Originate Call
    1 Jun  6 16:45:30  CSPF: computation result accepted
```

```
 *Secondary via-Chianti      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 16:49:39  Selected as active path
    4 Jun  6 16:49:39  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 16:49:39  Up
    2 Jun  6 16:49:39  Originate Call
    1 Jun  6 16:49:39  CSPF: computation result accepted
  Created: Fri Jun  6 16:45:20 2003
Total 1 displayed, Up 1, Down 0
```

Once the retry waiting period expires, the ingress router determines that the ***via-Chablis*** secondary path is not needed and it is cleared from the network. The active path for the LSP remains the ***via-Chianti*** secondary path, and we've successfully created a non-revertive LSP:

```
user@Sherry> show mpls lsp extensive ingress
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chianti (secondary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  Secondary via-Chablis      State: Dn
        No computed ERO.
   18 Jun  6 16:50:52  Clear Call
   17 Jun  6 16:49:54  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
   16 Jun  6 16:49:54  Up
   15 Jun  6 16:49:54  Originate Call
   14 Jun  6 16:49:54  CSPF: computation result accepted
   13 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
   12 Jun  6 16:49:39  Clear Call
   11 Jun  6 16:49:39  CSPF: link down/deleted 10.222.60.1(Chablis.00/
   10 Jun  6 16:49:39  Deselected as active
    9 Jun  6 16:49:39  ResvTear received
    8 Jun  6 16:49:39  Down
    7 Jun  6 16:49:39  CSPF failed: no route toward 192.168.32.1
    6 Jun  6 16:49:39  10.222.5.2: No Route toward dest
    5 Jun  6 16:45:30  Selected as active path
    4 Jun  6 16:45:30  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  6 16:45:30  Up
    2 Jun  6 16:45:30  Originate Call
    1 Jun  6 16:45:30  CSPF: computation result accepted
 *Secondary via-Chianti      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.29.2 S 10.222.1.2 S 10.222.45.2 S
    Received RRO:
          10.222.29.2 10.222.1.2 10.222.45.2
    5 Jun  6 16:49:39  Selected as active path
    4 Jun  6 16:49:39  Record Route:  10.222.29.2 10.222.1.2 10.222.45.2
    3 Jun  6 16:49:39  Up
    2 Jun  6 16:49:39  Originate Call
    1 Jun  6 16:49:39  CSPF: computation result accepted
  Created: Fri Jun  6 16:44:05 2003
Total 1 displayed, Up 1, Down 0
```

Fast Reroute

At this point in the chapter, we've talked about protecting traffic using secondary paths as well as paths in the standby state. These options work very well as long-term solutions in the network, but each has the potential for dropping packets in a failure mode. Specifically, these are the packets injected into the LSP by the ingress router between the time the failure occurs and the time the ingress is notified of the failure via a `ResvTear` message. For critical applications, this short failure time and its packet drops are unacceptable. The resolution to this situation is the use of *fast reroute* on an LSP. The basic operation of fast reroute involves the establishment of detour paths through the network that are associated with the RSVP session of the main LSP. Each router along the LSP's path creates a detour to protect against the failure of its next downstream neighbor or the next downstream link.

During a failure mode, the next upstream router from the point of failure immediately forwards traffic for the LSP along the detour path. This quick decision allows for minimal packet loss, if any occurs at all. This node also generates a `PathErr` message and forwards it to the ingress router. The error message is designed to alert the ingress that a failure occurred along the path and that the detour is being used to forward traffic.

The action taken by the ingress router then depends on its configuration and the state of the network. If a standby secondary path is established, the ingress router begins using it to forward traffic. The primary path, along with the detours, are torn down and the ingress attempts to locate another primary path through the network matching the LSP constraints. This same process occurs when a secondary path is configured but is not in an `Up` state. The ingress router signals and establishes the secondary path, moves traffic onto it, and tears down the primary path. If the ingress router has no secondary paths defined, it examines the TED to locate a new primary path meeting the constraints. If one is found, the ingress router creates a new primary path and moves the traffic onto it. This process occurs in one of two fashions:; either the ingress router creates the new path, moves traffic, and tears down the old path or the ingress tears down the old path, creates the new path, and moves the traffic. The actions of the ingress router in this situation are controlled by the `adaptive` command, which we discuss in the "Controlling LSP Behavior" section, later in this chapter. The ingress router might encounter one final possibility during a failure mode. There may be no secondary paths defined and the CSPF algorithm might not find a new primary path that meets the LSP constraints. When this occurs, the ingress router continues using the existing path and the detours to forward traffic across the network.

The JUNOS software performs fast reroute actions in one of two modes—node protection and link protection. Let's explore each in further detail.

Node Protection

In a *node protection* fast reroute environment, each router along the LSP path builds a detour path from itself to the egress router. This path is calculated using the information in the TED, and a strict-hop ERO is created for the detour. By default, the detour path inherits the administrative group settings of the main LSP to provide a limit as to which links the detour may use. In addition, you can apply a bandwidth reservation to the detour as well as a limit on the number of hops the detour can use. The main purpose of the detour path is to protect against the failure of the next downstream node in the LSP path. By default, this protection scheme also protects against the network link connecting the two neighbors. You enable fast reroute node protection with the `fast-reroute` command within the LSP itself:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
fast-reroute;
primary via-Chablis;
```

FIGURE 8.7 Fast reroute sample network

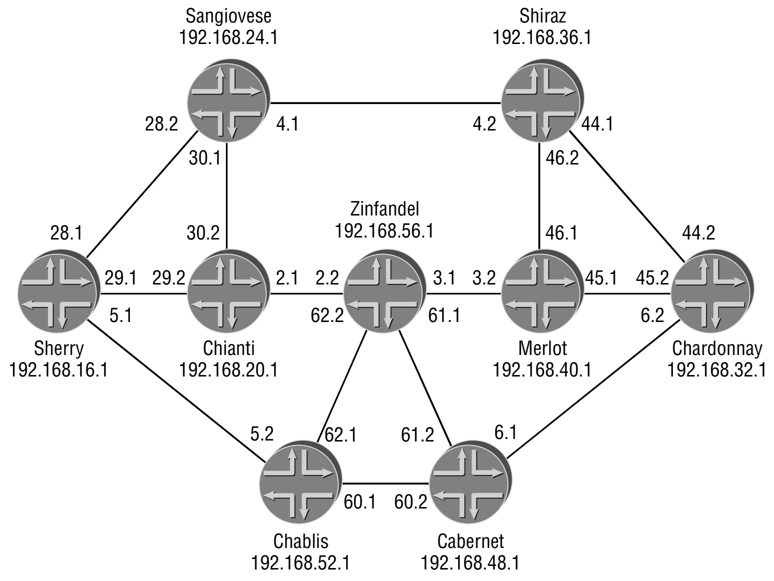


Figure 8.7 displays the network used to construct the ***Sherry-to-Char*** LSP. After the configuration we just described is committed to the router, Sherry begins sending `Path` messages along the LSP path that contain the Fast Reroute object. This object alerts each node along the path that it should create a detour path once the main LSP is established in the network. By examining the output of the `show mpls lsp extensive` command, we can see the establishment of the detour paths from the ingress router:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Chablis (primary)
  FastReroute desired
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Chablis      State: Up
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.5.2 S 10.222.60.2 S 10.222.6.2 S
    Received RRO:
          10.222.5.2(flag=0x9) 10.222.60.2(flag=0x1) 10.222.6.2
    8 Jun  7 11:26:46  Record Route:  10.222.5.2(flag=0x9) 10.222.60.2(flag=0x1)
    7 Jun  7 11:26:46  Record Route:  10.222.5.2(flag=0x9) 10.222.60.2 10.222.6.
    6 Jun  7 11:26:46  Fast-reroute Detour Up
    5 Jun  7 11:26:43  Selected as active path
    4 Jun  7 11:26:43  Record Route:  10.222.5.2 10.222.60.2 10.222.6.2
    3 Jun  7 11:26:43  Up
    2 Jun  7 11:26:43  Originate Call
    1 Jun  7 11:26:43  CSPF: computation result accepted
  Created: Sat Jun  7 11:26:37 2003
Total 1 displayed, Up 1, Down 0
```

NOTE Some lines in the output extend beyond 80 characters and have been truncated for readability.

Within the attributes of the LSP itself, the `FastReroute desired` output tells us that the object was included in the `Path` messages for the LSP. This allows Sherry to select the primary path as the active path for the LSP and then establish a detour path to avoid Chablis. While we don't know how the detour was established, we can see that it is operational through the `Fast-reroute Detour Up` output. The next two entries in the history buffer inform us that the two transit routers, Chablis and Cabernet, have also established their detour paths. The Chablis router, 10.222.5.2, includes the `flag=0x9` notation in its Record Route object (RRO) within the `Resv` message. This informs the upstream routers that a successful detour is established that protects both the downstream node and the downstream link. The Cabernet router, 10.222.6.2, includes the `flag=0x1` notation in its RRO, which states that a detour is established to protect the next downstream link. While it might seem unusual for Cabernet to not protect the downstream node, a quick look at the network map reveals that the egress router of Chardonnay is the next downstream node. It is impossible for Cabernet to avoid the egress router, so only the downstream link is protected by the Cabernet router.

To see the actual links used by the detour paths, we need to use the `show rsvp session detail` command. As we move around the network, this command provides us with complete knowledge of the main LSP and all the detour paths in the network. From the perspective of the Sherry router we see the following:

```
user@Sherry> show rsvp session ingress detail
Ingress RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100256
  Resv style: 1 FF, Label in: -, Label out: 100256
  Time left:    -,  Since: Sat Jun  7 11:26:43 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  FastReroute desired
  PATH rcvfrom: localclient
  PATH sentto: 10.222.5.2 (at-0/1/1.0) 56 pkts
  RESV rcvfrom: 10.222.5.2  (at-0/1/1.0) 58 pkts
  Explct route: 10.222.5.2 10.222.60.2 10.222.6.2
  Record route: <self>  10.222.5.2  10.222.60.2  10.222.6.2
    Detour is Up
    Detour PATH sentto: 10.222.28.2 (at-0/1/0.0) 55 pkts
    Detour RESV rcvfrom: 10.222.28.2  (at-0/1/0.0) 55 pkts
    Detour Explct route: 10.222.28.2 10.222.4.2 10.222.44.2
    Detour Record route: <self>  10.222.28.2  10.222.4.2  10.222.44.2
    Detour Label out: 100240
Total 1 displayed, Up 1, Down 0
```

Within the context of the RSVP session representing the main LSP, we see that the detour path is established by the `Detour is Up` notation. The physical path of the detour is displayed by the `Detour Explct route:` output and uses Sangiovese and Shiraz as transit routers before reaching the egress router of Chardonnay. When we examine the session data on the first transit router of Chablis, we see some similar information:

```
user@Chablis> show rsvp session transit detail
Transit RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100240
  Resv style: 1 FF, Label in: 100256, Label out: 100240
  Time left:  140,  Since: Fri Sep 28 13:54:33 2001
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  FastReroute desired
  PATH rcvfrom: 10.222.5.1  (at-0/2/0.0) 237 pkts
  PATH sentto: 10.222.60.2 (so-0/1/0.0) 237 pkts
  RESV rcvfrom: 10.222.60.2  (so-0/1/0.0) 238 pkts
  Explct route: 10.222.60.2 10.222.6.2
  Record route: 10.222.5.1  <self>  10.222.60.2  10.222.6.2
    Detour is Up
    Detour PATH sentto: 10.222.62.2 (so-0/1/1.0) 236 pkts
    Detour RESV rcvfrom: 10.222.62.2  (so-0/1/1.0) 236 pkts
    Detour Explct route: 10.222.62.2 10.222.3.2 10.222.45.2
    Detour Record route: 10.222.5.1  <self>  10.222.62.2  10.222.3.2
    10.222.45.2
    Detour Label out: 100176
Total 1 displayed, Up 1, Down 0
```

The detour created by Chablis is designed to avoid the Cabernet router as well as the link connecting Chablis to Cabernet. The detour is currently in an `Up` state and uses Zinfandel and Merlot as transit routers before reaching the egress router of Chardonnay. In fact, this is the exact same detour path used by Cabernet to avoid its downstream link to the egress router:

```
user@Cabernet> show rsvp session transit detail
Transit RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 3
  Resv style: 1 FF, Label in: 100240, Label out: 3
  Time left:  126,  Since: Sat Jan  5 13:59:04 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  FastReroute desired
  PATH rcvfrom: 10.222.60.1  (so-0/1/2.0) 243 pkts
  PATH sentto: 10.222.6.2 (so-0/1/1.0) 243 pkts
  RESV rcvfrom: 10.222.6.2  (so-0/1/1.0) 243 pkts
  Explct route: 10.222.6.2
  Record route: 10.222.5.1  10.222.60.1  <self>  10.222.6.2
    Detour is Up
    Detour PATH sentto: 10.222.61.1 (so-0/1/0.0) 242 pkts
    Detour RESV rcvfrom: 10.222.61.1  (so-0/1/0.0) 242 pkts
    Detour Explct route: 10.222.61.1 10.222.3.2 10.222.45.2
    Detour Record route: 10.222.5.1  10.222.60.1  <self>  10.222.61.1
    10.222.3.2  10.222.45.2
    Detour Label out: 100160
Total 1 displayed, Up 1, Down 0
```

Because the detour paths from Chablis and Cabernet use Zinfandel as a transit router, the fast reroute specification allows Zinfandel to merge the detour paths together. After all, they belong to the same main RSVP session and they are both sending packets to the same egress router. So, there's no harm in combining them into a single flow between Zinfandel and Chardonnay. As we look at the `show rsvp session transit detail` output on Zinfandel, we see the merge operation take place:

```
user@Zinfandel> show rsvp session transit detail
Transit RSVP: 1 sessions, 1 detours

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100224
  Resv style: 1 FF, Label in: 100176, Label out: 100224
  Time left:  120,  Since: Sat Jun  7 11:26:16 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  Detour branch from 10.222.62.1, to skip 192.168.48.1, Up
    PATH rcvfrom: 10.222.62.1  (so-0/1/3.0) 247 pkts
    PATH sentto: 10.222.3.2 (so-0/1/1.0) 247 pkts
    RESV rcvfrom: 10.222.3.2  (so-0/1/1.0) 248 pkts
    Explct route: 10.222.3.2 10.222.45.2
    Record route: 10.222.5.1  10.222.62.1  <self>  10.222.3.2  10.222.45.2
    Label in: 100176, Label out: 100224
  Detour branch from 10.222.61.2, to skip 192.168.32.1, Up
    PATH rcvfrom: 10.222.61.2  (so-0/1/2.0) 247 pkts
    PATH sentto: 10.222.3.2 (so-0/1/1.0) 0 pkts
    RESV rcvfrom: 10.222.3.2  (so-0/1/1.0) 0 pkts
    Explct route: 10.222.3.2 10.222.45.2
    Record route: 10.222.5.1  10.222.60.1  10.222.61.2  <self>  10.222.3.2
    10.222.45.2
    Label in: 100160, Label out: 100224
Total 1 displayed, Up 1, Down 0
```

The first thing to note is that both detours appear without the output of the same session, which includes the ingress address, the egress address, and the name of the LSP. We then see the detour path arrive from the Chablis router as noted by the `Detour branch from 10.222.62.1` output. This also informs us that Chablis is attempting to avoid Cabernet (`to skip 192.168.48.1`). A similar set of data appears from Cabernet (10.222.61.2) to avoid the link connecting it to the egress router (192.168.32.1). The actual merge functionality that Zinfandel is performing can be seen by the label used to send traffic along the detour. Both detours have the same `Label out: 100224` notation identified. This means that whether Zinfandel receives traffic from Chablis with a label of 100,176 or from Cabernet with a label of 100,160, it forwards the packet to Merlot with a label value of 100,224. When we look at Merlot's view of the detour, we see only a single set of information:

```
user@Merlot> show rsvp session transit detail
Transit RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 3
  Resv style: 1 FF, Label in: 100224, Label out: 3
  Time left:  140,  Since: Sat Jun  7 11:27:31 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  Detour branch from 10.222.62.1, to skip 192.168.48.1, Up
  Detour branch from 10.222.61.2, to skip 192.168.32.1, Up
    PATH rcvfrom: 10.222.3.1  (so-0/1/1.0) 257 pkts
    PATH sentto: 10.222.45.2 (so-0/1/2.0) 259 pkts
    RESV rcvfrom: 10.222.45.2  (so-0/1/2.0) 257 pkts
    Explct route: 10.222.45.2
    Record route: 10.222.5.1  10.222.62.1  10.222.3.1  <self>  10.222.45.2
    Label in: 100224, Label out: 3
Total 1 displayed, Up 1, Down 0
```

Both the ERO of the detour and the `Label out:` information tell us that Merlot is the penultimate router for the detour path. We can tell that the detour was merged at some point upstream since multiple `Detour branch` statements appear before the `Path` and `Resv` message information.

When we finally reach the egress router of Chardonnay, we see information pertaining to the main LSP path as well as the two detours through the network:

```
user@Chardonnay> show rsvp session egress detail
Egress RSVP: 1 sessions, 2 detours

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: -
  Resv style: 1 FF, Label in: 3, Label out: -
  Time left:  146,  Since: Sat Jun  7 11:25:20 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11185 protocol 0
  FastReroute desired
  PATH rcvfrom: 10.222.6.1  (so-0/1/2.0) 263 pkts
  PATH sentto: localclient
  RESV rcvfrom: localclient
  Record route: 10.222.5.1  10.222.60.1  10.222.6.1  <self>
  Detour branch from 10.222.28.1, to skip 192.168.52.1, Up
    PATH rcvfrom: 10.222.44.1  (so-0/1/0.0) 262 pkts
    PATH sentto: localclient
    RESV rcvfrom: localclient
    Record route: 10.222.28.1  10.222.4.1  10.222.44.1  <self>
    Label in: 3, Label out: -
  Detour branch from 10.222.62.1, to skip 192.168.48.1, Up
  Detour branch from 10.222.61.2, to skip 192.168.32.1, Up
    PATH rcvfrom: 10.222.45.1  (so-0/1/1.0) 264 pkts
    PATH sentto: localclient
    RESV rcvfrom: localclient
    Record route: 10.222.5.1  10.222.62.1  10.222.3.1  10.222.45.1  <self>
    Label in: 3, Label out: -
Total 1 displayed, Up 1, Down 0
```

Link Protection

Although the purpose of *link protection* fast reroute remains the continued forwarding of traffic using MPLS in a failure mode, the actual mechanics used by link protection are quite different. When we examined a node-protection scheme in the previous section, we found that each node along the LSP generated a detour path that was associated with the main RSVP session. During a failure, the next upstream router, officially referred to as the *point of local repair* (PLR), performed a label swap operation for all incoming data packets and forwarded them along the detour path. In a link-protection environment, we create bypass LSPs from one router to its neighbor that avoid the interconnecting link. This is a function of RSVP and is enabled through the `link-protection` command on each RSVP interface requiring protection. The bypass LSPs are established when the ingress router for the main LSP configures the `link-protection` command to request that each node along the path use the bypass LSPs to protect the traffic flow.

When a failure occurs, the PLR performs a label swap operation to place the label advertised by the downstream node in the bottom MPLS header. The PLR then performs a label push operation to add the label corresponding to the bypass LSP and forwards the packet along the bypass. The penultimate router along the bypass pops the top label value and forwards the remaining data to the router that is downstream of the point of local repair. The received label value is known to this router since it originally allocated it for the main LSP. As such, it performs the appropriate label function and forwards the traffic along the path of the main LSP. Through this process, a bypass LSP can officially support repair capabilities for multiple LSPs in a many-to-one fashion.

Using the network in Figure 8.7, we first verify that the Cabernet router has no current knowledge of any LSPs or RSVP sessions:

```
user@Cabernet> 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: 0 sessions
Total 0 displayed, Up 0, Down 0

user@Cabernet> show rsvp session
Ingress RSVP: 0 sessions
Total 0 displayed, Up 0, Down 0

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

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

We then configure each of the interfaces on Cabernet for link protection, which establishes bypass LSPs to protect the interfaces to Chablis, Zinfandel, and Chardonnay:

```
user@Cabernet> show configuration protocols rsvp
interface all {
    link-protection;
}

user@Cabernet> show rsvp session ingress detail
Ingress RSVP: 3 sessions

192.168.32.1
  From: 192.168.48.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass_to_10.222.6.2
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100240
  Resv style: 1 SE, Label in: -, Label out: 100240
  Time left:    -,  Since: Sat Jan  5 18:06:34 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 4464 protocol 0
  Type: Bypass LSP
  PATH rcvfrom: localclient
  PATH sentto: 10.222.61.1 (so-0/1/0.0) 4 pkts
  RESV rcvfrom: 10.222.61.1  (so-0/1/0.0) 4 pkts
  Explct route: 10.222.61.1 10.222.3.2 10.222.45.2
  Record route: <self>  10.222.61.1  10.222.3.2  10.222.45.2

192.168.52.1
  From: 192.168.48.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass_to_10.222.60.1
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100224
  Resv style: 1 SE, Label in: -, Label out: 100224
  Time left:    -,  Since: Sat Jan  5 18:06:34 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 4465 protocol 0
  Type: Bypass LSP
  PATH rcvfrom: localclient
  PATH sentto: 10.222.61.1 (so-0/1/0.0) 4 pkts
  RESV rcvfrom: 10.222.61.1  (so-0/1/0.0) 4 pkts
  Explct route: 10.222.61.1 10.222.62.1
  Record route: <self>  10.222.61.1  10.222.62.1

192.168.56.1
  From: 192.168.48.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass_to_10.222.61.1
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100288
  Resv style: 1 SE, Label in: -, Label out: 100288
  Time left:    -,  Since: Sat Jan  5 18:06:34 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 4463 protocol 0
  Type: Bypass LSP
  PATH rcvfrom: localclient
  PATH sentto: 10.222.60.1 (so-0/1/2.0) 4 pkts
  RESV rcvfrom: 10.222.60.1  (so-0/1/2.0) 4 pkts
  Explct route: 10.222.60.1 10.222.62.2
  Record route: <self>  10.222.60.1  10.222.62.2
Total 3 displayed, Up 3, Down 0
```

The first critical piece of information to note is the creation of three separate RSVP ingress sessions, one for each interface. Each session is described as a bypass LSP via the `Type:` output, and each has an LSP name automatically created for it. By default, the name appears as `Bypass_to_`*`interface-address`*, where the interface address is the next downstream router's interface. For example, Cabernet established the ***Bypass_to_10.222.61.1*** LSP between itself and Zinfandel (192.168.56.1) to protect their interconnected network link. The `Explct route` for the session displays Chablis as the transit node for the bypass and Zinfandel as the egress node.

To take advantage of the established link-protection bypasses, the ***Sherry-to-Char*** LSP also configures the `link-protection` command:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
link-protection;
primary via-Chablis;
```

This adds an RSVP transit session on the Cabernet router:

NOTE For ease in interpreting the output, only the outgoing interfaces along the ***Sherry-to-Char*** path have been configured with `link-protection`.

```
user@Cabernet> show rsvp session detail
Ingress RSVP: 1 sessions
```

```
192.168.32.1
  From: 192.168.48.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Bypass_to_10.222.6.2
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100272
  Resv style: 1 SE, Label in: -, Label out: 100272
  Time left:    -,  Since: Sat Jan  5 18:19:23 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 4466 protocol 0
  Type: Bypass LSP
  PATH rcvfrom: localclient
  PATH sentto: 10.222.61.1 (so-0/1/0.0) 5 pkts
  RESV rcvfrom: 10.222.61.1  (so-0/1/0.0) 5 pkts
  Explct route: 10.222.61.1 10.222.3.2 10.222.45.2
  Record route: <self>  10.222.61.1  10.222.3.2  10.222.45.2
Total 1 displayed, Up 1, Down 0

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

Transit RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 3
  Resv style: 1 SE, Label in: 100288, Label out: 3
  Time left:  156,  Since: Sat Jan  5 18:20:19 2002
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 3 receiver 11196 protocol 0
  Link protection desired
  Type: Link protected LSP
  PATH rcvfrom: 10.222.60.1  (so-0/1/2.0) 3 pkts
  PATH sentto: 10.222.6.2 (so-0/1/1.0) 3 pkts
  RESV rcvfrom: 10.222.6.2  (so-0/1/1.0) 4 pkts
  Explct route: 10.222.6.2
  Record route: 10.222.5.1  10.222.60.1  <self>  10.222.6.2
Total 1 displayed, Up 1, Down 0
```

Within the transit RSVP section for the ***Sherry-to-Char*** LSP, we see the `Link protection desired` output requesting protection from the ingress router. Since Cabernet has a bypass LSP in place to protect the downstream link, it associates this LSP with the bypass. This is indicated by the `Type: Link protected LSP` notation.

When we examine the output of this command on the transit router of Chablis, we see different information. Although the LSP is requesting link protection, no bypass LSP is established from Chablis to any of its neighbors. As such, the LSP is not actually link protected:

```
user@Chablis> show rsvp session detail
Ingress RSVP: 0 sessions
Total 0 displayed, Up 0, Down 0

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

Transit RSVP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 100288
  Resv style: 1 SE, Label in: 100336, Label out: 100288
  Time left:  122,  Since: Fri Sep 28 18:15:48 2001
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 3 receiver 11196 protocol 0
  Link protection desired
  PATH rcvfrom: 10.222.5.1  (at-0/2/0.0) 10 pkts
  PATH sentto: 10.222.60.2 (so-0/1/0.0) 10 pkts
  RESV rcvfrom: 10.222.60.2  (so-0/1/0.0) 10 pkts
  Explct route: 10.222.60.2 10.222.6.2
  Record route: 10.222.5.1  <self>  10.222.60.2  10.222.6.2
Total 1 displayed, Up 1, Down 0
```

Controlling LSP Behavior

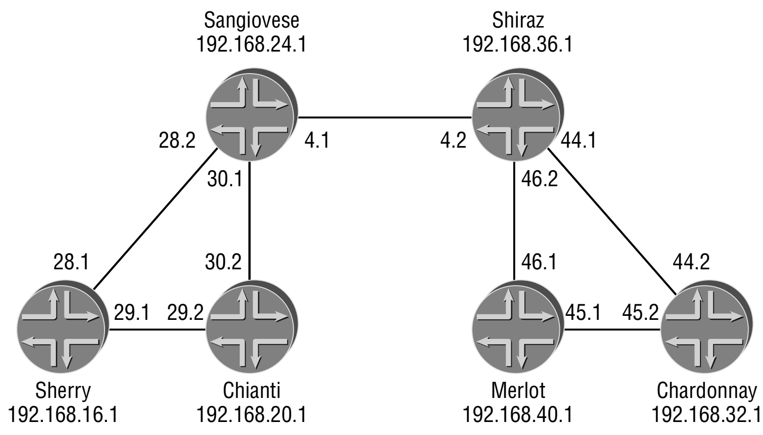
The JUNOS software also provides you with various methods for controlling how your LSPs establish themselves and operate within your network. These include the sharing of bandwidth reservations between paths of the same LSP, the advertisement of a 0 label by the egress router, the ability to hide the physical connectivity of your network, and the use of your LSPs for forwarding traffic for non-BGP learned routes. While I would agree that these topics don't necessarily have a natural fit with one another, they are important nonetheless. So, let's discuss each in turn.

Adaptive Mode

The default bandwidth reservation style in the JUNOS software is *fixed filter* (FF). This style allocates a unique bandwidth reservation for each sender (LSP ID) and receiver (Tunnel ID) pair. The opposite reservation style is known as *shared explicit* (SE), which allocates a single reservation for each unique RSVP session. This reservation can be shared among multiple sending ID values from the ingress router. The SE reservation style is enabled by configuring the `adaptive` keyword within your LSP.

Using *adaptive mode* in your network provides you with two unique qualities. The first of these is the establishment of a standby secondary path, which shares physical links in common with the LSP's primary path. As an example, suppose that an LSP is requesting 75Mbps for a bandwidth reservation for both the primary and secondary paths. At some point between the ingress and egress router, the two paths must share a Fast Ethernet segment that has not been oversubscribed. When the primary path is established in the network using the default FF reservation style, the 75Mbps reservation on the Fast Ethernet link is allocated solely for its use. This means that the ingress router can't establish the secondary path of the LSP since the available bandwidth of the Fast Ethernet is below 75Mbps. Such an example is shown in Figure 8.8.

FIGURE 8.8 Adaptive mode



The ***Sherry-to-Char*** LSP is configured for a 75Mbps bandwidth reservation. The primary path is configured to use Sangiovese as a transit router, while the secondary path is configured to use Chianti as a transit router. The physical link between the Sangiovese and Shiraz routers is a Fast Ethernet segment that hasn't been oversubscribed. To allow both the primary and secondary paths to be established, the `adaptive` keyword is configured within the LSP:

```
[edit protocols mpls]
user@Sherry# show label-switched-path Sherry-to-Char
to 192.168.32.1;
```

```
bandwidth 75m;
adaptive;
primary via-Sangiovese;
secondary via-Chianti {
    standby;
}
```

WARNING

The placement of the `adaptive` keyword at the global LSP level is quite critical here. This is what keeps the RSVP session information the same for the primary and secondary paths. If the `adaptive` command is placed within the primary and secondary paths themselves, the Tunnel ID values are not identical. This causes the paths to be viewed as separate RSVP sessions, which may not share the same bandwidth reservation.

Both the primary and secondary paths are established in the network and are in an `Up` state:

```
user@Sherry> show mpls lsp ingress extensive
Ingress LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, State: Up, ActiveRoute: 0, LSPname: Sherry-to-Char
  ActivePath: via-Sangiovese (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary   via-Sangiovese   State: Up
    Bandwidth: 75Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 30)
          10.222.28.2 S 10.222.4.2 S 10.222.44.2 S
    Received RRO:
          10.222.28.2 10.222.4.2 10.222.44.2
    6 Jun  7 18:01:18  Selected as active path
    5 Jun  7 18:01:18  Record Route:  10.222.28.2 10.222.4.2 10.222.44.2
    4 Jun  7 18:01:18  Up
    3 Jun  7 18:01:18  Originate Call
    2 Jun  7 18:01:18  CSPF: computation result accepted
    1 Jun  7 18:00:49  CSPF failed: no route toward 192.168.32.1
  Standby   via-Chianti      State: Up
    Bandwidth: 75Mbps
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 40)
          10.222.29.2 S 10.222.30.1 S 10.222.4.2 S 10.222.44.2 S
    Received RRO:
          10.222.29.2 10.222.30.1 10.222.4.2 10.222.44.2
    4 Jun  7 18:01:18  Record Route:  10.222.29.2 10.222.30.1 10.222.4.2 10.222
    3 Jun  7 18:01:18  Up
    2 Jun  7 18:01:18  Originate Call
    1 Jun  7 18:01:18  CSPF: computation result accepted
  Created: Sat Jun  7 18:00:32 2003
Total 1 displayed, Up 1, Down 0
```

While both the paths are using the Sangiovese-Shiraz link, only a single 75Mbps bandwidth reservation is in place:

```
user@Sangiovese> show rsvp interface
RSVP interface: 3 active
                  Active Subscr- Static      Available   Reserved    Highwater
Interface   State resv   iption  BW          BW          BW          mark
fe-0/0/2.0  Up         1   100%  100Mbps     25Mbps      75Mbps      90Mbps
at-0/1/0.0  Up         0   100%  155.52Mbps  155.52Mbps  0bps        0bps
so-0/2/0.0  Up         0   100%  155.52Mbps  155.52Mbps  0bps        0bps
```

The second major advantage to using adaptive mode comes into play when an established path attempts to reroute itself onto new network links. In this situation, the `adaptive` setting allows the ingress router to establish the new path and move traffic to it before removing the old path from the network. This action is referred to as a *make-before-break* operation. In addition, adaptive mode allows the establishing path to not double-count bandwidth reservations on links shared with the path being torn down.

Explicit Null Advertisements

Before we discuss the advertisement of the explicit null label by the egress router, it is helpful to quickly talk about how class of service is handled within the JUNOS software. Within the scope of a single router, the information in the packet header determines how the packet is queued on the outbound interface. In the default operating environment, the ingress router classifies the incoming packet and an outbound queue is determined. The ingress router may also specify the value of the experimental bits in the MPLS header, which allows the first transit router to classify and queue the packet differently. This process continues until the penultimate router is reached. In this case, the incoming packet contains an MPLS header and the experimental bits in that header classify the packet. Since the penultimate router was signaled a label value of 3, the implicit null value, the penultimate router pops the MPLS header and forwards the native IPv4 packet to the egress router. The egress router then performs its class of service operations on the IP Precedence bits in the IP packet. The contents of these bits may or may not allow for the proper class of service processing on the egress router.

In this situation, it may be advantageous to allow the egress router to advertise a label value of 0, the explicit null value, to the penultimate router. This allows the penultimate router to forward packets to the egress router with an MPLS header attached. Assuming that each router in the network has maintained the settings within the experimental bits, the egress router now has the ability to classify the packet using information set by the ingress router.

You enable this functionality by configuring the `explicit-null` command at the global MPLS level on the egress router. We see this configuration on the Chardonnay router in Figure 8.8:

```
user@Chardonnay> show configuration protocols mpls
explicit-null;
interface all;
```

When the ***Sherry-to-Char*** LSP is established in the network, Chardonnay signals a label value of 0 to Shiraz. In addition to ensuring that the LSP is established, we check on Shiraz and Chardonnay to verify that the correct label was advertised:

```
user@Sherry> show mpls lsp ingress
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.16.1    Up     0                  *     Sherry-to-Char
Total 1 displayed, Up 1, Down 0

user@Shiraz> show mpls lsp transit detail
Transit LSP: 1 sessions

192.168.32.1
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 1
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 0
  Resv style: 1 FF, Label in: 100384, Label out: 0
  Time left:  156,  Since: Sun Jun  8 17:59:03 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11216 protocol 0
  PATH rcvfrom: 10.222.4.1  (fe-0/0/2.0) 4 pkts
  PATH sentto: 10.222.44.2 (so-0/1/0.0) 4 pkts
  RESV rcvfrom: 10.222.44.2  (so-0/1/0.0) 4 pkts
  Explct route: 10.222.44.2
  Record route: 10.222.28.1  10.222.4.1  <self>  10.222.44.2
Total 1 displayed, Up 1, Down 0

user@Chardonnay> show mpls lsp egress detail
Egress LSP: 1 sessions

192.168.32.1
```

```
  From: 192.168.16.1, LSPstate: Up, ActiveRoute: 0
  LSPname: Sherry-to-Char, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: -
  Resv style: 1 FF, Label in: 0, Label out: -
  Time left:  146,  Since: Sun Jun  8 18:01:18 2003
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 11216 protocol 0
  PATH rcvfrom: 10.222.44.1  (so-0/1/0.0) 4 pkts
  PATH sentto: localclient
  RESV rcvfrom: localclient
  Record route: 10.222.28.1  10.222.4.1  10.222.44.1  <self>
Total 1 displayed, Up 1, Down 0
```

Controlling Time-to-Live

The default operation of the JUNOS software with respect to time-to-live (TTL) is to copy the value from the IP packet to the MPLS header after decrementing it b y 1. The LSP routers decrement the value by one at each hop, and the value is rewritten to the IP header when the label is popped by the penultimate router. This default operation allows each hop in the LSP's path to become visible to devices outside the MPLS domain.

FIGURE 8.9 Time-to-live sample network

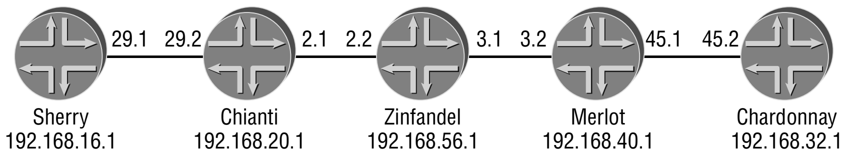


Figure 8.9 displays a network with the Chianti, Zinfandel, and Merlot routers participating in an MPLS network. The ***Chianti-to-Merlot*** LSP is established and operational in the network:

```
user@Chianti> show mpls lsp ingress
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P    LSPname
192.168.40.1    192.168.20.1    Up     1                  *    Chianti-to-Merlot
Total 1 displayed, Up 1, Down 0
```

The Sherry router is receiving the 192.168.32.0 /24 route via BGP, which allows reachability to the loopback address of Chardonnay:

```
user@Sherry> show route protocol bgp terse
```

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 192.168.32.0/24    B 170        100            >10.222.29.2     2 3 I
```

When the Sherry router performs a `traceroute` to a host within that subnet, each of the MPLS routers is visible in the output:

```
user@Sherry> traceroute 192.168.32.10 source 192.168.16.1
traceroute to 192.168.32.10 (192.168.32.10) from 192.168.16.1, 30 hops max
 1  10.222.29.2 (10.222.29.2)  0.731 ms  0.547 ms  0.488 ms
 2  10.222.2.2 (10.222.2.2)  0.792 ms  0.730 ms  0.704 ms
     MPLS Label=100288 CoS=0 TTL=1 S=1
 3  10.222.3.2 (10.222.3.2)  0.575 ms  0.563 ms  0.538 ms
 4  10.222.45.2 (10.222.45.2)  0.604 ms !N  0.584 ms !N  0.560 ms !N
```

Using the *no-decrement-ttl* Command

The first option available in the JUNOS software for hiding the details of an LSP's path is the `no-decrement-ttl` command. This option requires each router in the LSP to understand a special Label Request object, which signals the ingress router's desire to alter the default TTL processing rules. Once the LSP is established, the ingress router doesn't copy the IP TTL value into the MPLS header. Instead, it places a value of 255 in the MPLS header and forwards the packet to the first transit router. When the packet arrives at the penultimate router, the value in the MPLS header is not written back to the IP packet. Instead, the top label is popped and the remaining packet data is forwarded to the egress router.

The ***Chianti-to-Merlot*** LSP in Figure 8.9 is configured with the `no-decrement-ttl` command. This establishes the LSP in the network where we see the results in the output of the `show mpls lsp detail` command:

```
user@Chianti> show mpls lsp ingress detail
Ingress LSP: 1 sessions

192.168.40.1
  From: 192.168.20.1, State: Up, ActiveRoute: 1, LSPname: Chianti-to-Merlot
  ActivePath:  (primary)
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up, No-decrement-ttl
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 20)
          10.222.2.2 S 10.222.3.2 S
```

```
   Received RRO:
         10.222.2.2 10.222.3.2
Total 1 displayed, Up 1, Down 0
```

When we trace the route to the 192.168.32.0 /24 subnet from the Sherry router, we lose our ability to view some details of the LSP path:

```
user@Sherry> traceroute 192.168.32.10 source 192.168.16.1
traceroute to 192.168.32.10 (192.168.32.10) from 192.168.16.1, 30 hops max
 1  10.222.29.2 (10.222.29.2)  0.751 ms  0.552 ms  0.480 ms
 2  10.222.3.2 (10.222.3.2)  0.574 ms  0.563 ms  0.538 ms
 3  10.222.45.2 (10.222.45.2)  0.627 ms !N  9.542 ms !N  0.563 ms !N
```

More specifically, the Zinfandel router doesn't appear in the router output. Since the LSP is using penultimate hop popping, the Merlot router receives a native IPv4 packet. The TTL of the IP packet is set to a value of 1 as the ingress router decremented it prior to adding the MPLS header. This prompts Merlot to discard the packet and return an ICMP message back to the source of the packet.

Using the *no-propagate-ttl* Command

The second option available to hide the details of an LSP's path is the `no-propagate-ttl` command, which is configured at the `[edit protocols mpls]` global hierarchy level. This option doesn't signal any information in a `Path` message, which makes it interoperable with all other router vendors. As before, the ingress router doesn't copy the IP TTL value into the MPLS header and instead puts a 255 value in its place. When the packet arrives at the penultimate router, the TTL value is not written back into the IP packet, which is forwarded natively to the egress router.

Each of the routers in the path of the ***Chianti-to-Merlot*** LSP is configured with the `no-propagate-ttl` command. The configuration on Chianti looks like this:

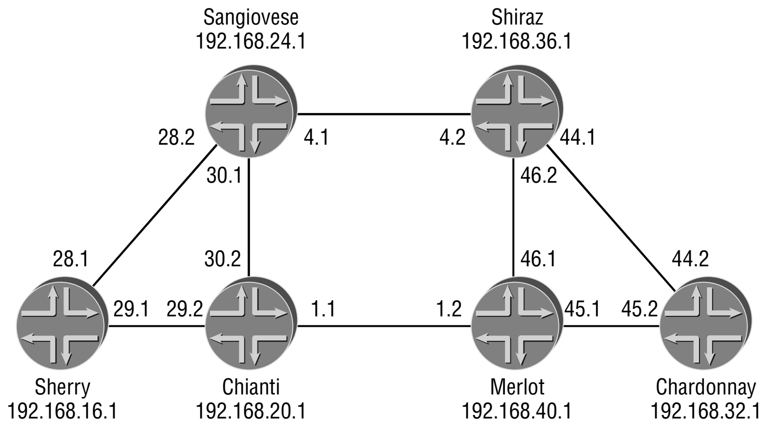
```
user@Chianti> show configuration protocols mpls
no-propagate-ttl;
label-switched-path Chianti-to-Merlot {
    to 192.168.40.1;
}
interface all;
```

When we trace the route to 192.168.32.1 with this configuration, we no longer see the egress router in the path:

```
user@Sherry> traceroute 192.168.32.10 source 192.168.16.1
traceroute to 192.168.32.10 (192.168.32.10) from 192.168.16.1, 30 hops max
 1  10.222.29.2 (10.222.29.2)  0.735 ms  0.542 ms  0.496 ms
 2  10.222.45.2 (10.222.45.2)  0.605 ms !N  0.580 ms !N  0.557 ms !N
```

This output is caused by the ingress router not decrementing the IP TTL before pushing the MPLS header onto the packet. When the egress router receives the native IPv4 packet, the TTL is set to a value of 2. Merlot routes the IP packet to Chardonnay, which is the end device in the path.

FIGURE 8.10 Controlling LSP interactions



LSP and Routing Protocol Interactions

The default use of an established LSP is the forwarding of BGP transit traffic across your domain. The router accomplishes this by matching the egress address of the LSP, in the `inet.3` routing table, to the address specified in the BGP Next Hop attribute. The JUNOS software provides multiple methods for altering or controlling this default selection process.

Advertising an LSP to the IGP

One convenient method for allowing the domain's IGP to forward traffic over an LSP is to advertise the connection into the link-state database as a point-to-point link. This *forwarding adjacency* is advertised within the routing advertisement by the ingress router with a defined metric value. As with a regular IGP adjacency, a bidirectional relationship must be established between the ingress and egress routers. As such, an LSP needs to be established in the return direction for the forwarding adjacency to become active.

Two LSPs are established between the Sangiovese and Chardonnay routers within the network in Figure 8.10. One LSP in each direction easily solves our bidirectional requirement:

```
user@Sangiovese> show mpls lsp
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.32.1    192.168.24.1    Up     0                  *     Sangio-to-Char
Total 1 displayed, Up 1, Down 0

Egress LSP: 1 sessions
To              From            State Rt Style Labelin Labelout LSPname
```

```
192.168.24.1    192.168.32.1    Up      0  1 FF       3        - Char-to-Sangio
Total 1 displayed, Up 1, Down 0

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

Using the `label-switched-path` command within the IGP configuration, IS-IS in our case, we assign a metric value of 5 to the LSP. This allows Sangiovese to begin advertising the LSP as a point-to-point link within the IS-IS network:

```
user@Sangiovese> show configuration protocols isis
level 1 disable;
level 2 wide-metrics-only;
interface all;
interface fxp0.0 {
    disable;
}
label-switched-path Sangio-to-Char {
    level 2 metric 5;
}

user@Sangiovese> show isis adjacency
Interface             System         L State        Hold (secs) SNPA
Sangio-to-Char        Chardonnay     0 One-way                0
at-0/1/0.0            Sherry         2 Up                    19
fe-0/0/2.0            Shiraz         2 Up                     7  0:90:69:68:c8:2
so-0/2/0.0            Chianti        2 Up                    21

user@Sangiovese> show isis database Sangiovese.00-00 detail
IS-IS level 1 link-state database:

IS-IS level 2 link-state database:

Sangiovese.00-00  Sequence: 0x2bb, Checksum: 0xc97c, Lifetime: 1119 secs
   IS neighbor:                     Sherry.00  Metric:      10
   IS neighbor:                    Chianti.00  Metric:      10
   IS neighbor:                 Chardonnay.00  Metric:       5
   IS neighbor:                     Shiraz.02  Metric:      10
   IP prefix:                 10.222.4.0/24 Metric:      10 Internal Up
   IP prefix:                10.222.28.0/24 Metric:      10 Internal Up
   IP prefix:                10.222.30.0/24 Metric:      10 Internal Up
   IP prefix:               192.168.24.1/32 Metric:       0 Internal Up
```

Because only Sangiovese has advertised the forwarding adjacency, it is not currently usable in the network. The adjacency state between Sangiovese and Chardonnay is set to `One-way` and no IS-IS routes are using the LSP for forwarding traffic:

```
user@Sangiovese> show route protocol isis terse

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.44.0/24     I  18         20            >10.222.4.2
* 10.222.45.0/24     I  18         30            >10.222.4.2
                                                  10.222.30.2
* 10.222.46.0/24     I  18         20            >10.222.4.2
* 192.168.32.1/32    I  18         20            >10.222.4.2
* 192.168.36.1/32    I  18         10            >10.222.4.2
* 192.168.40.1/32    I  18         20             10.222.4.2
                                                 >10.222.30.2
```

NOTE: We know that the LSP is not being used since all metric values are multiples of 10 and the LSP is not listed as a valid next hop. In addition, some routes have been removed for readability.

Once the forwarding adjacency is advertised by Chardonnay, the adjacency changes to an `Up` state and the LSP is used for forwarding traffic in the network:

```
user@Sangiovese> show isis adjacency
Interface             System         L State        Hold (secs) SNPA
Sangio-to-Char        Chardonnay     2 Up                     0
at-0/1/0.0            Sherry         2 Up                    20
fe-0/0/2.0            Shiraz         2 Up                     7  0:90:69:68:c8:2
so-0/2/0.0            Chianti        2 Up                    26

user@Sangiovese> show route protocol isis terse

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

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.44.0/24     I  18         15            >10.222.4.2
* 10.222.45.0/24     I  18        15            >10.222.4.2
* 10.222.46.0/24     I  18        20            >10.222.4.2
* 192.168.32.1/32    I  18         5            >10.222.4.2
* 192.168.36.1/32    I  18        10            >10.222.4.2
* 192.168.40.1/32    I  18        15            >10.222.4.2
```

For further proof that the LSP is being used to forward traffic, we trace the route to Chardonnay's loopback address from the Sherry router:

```
user@Sherry> traceroute 192.168.32.1
traceroute to 192.168.32.1 (192.168.32.1), 30 hops max, 40 byte packets
 1  10.222.28.2 (10.222.28.2)  1.232 ms  0.917 ms  0.896 ms
 2  10.222.4.2 (10.222.4.2)  0.993 ms  0.841 ms  0.917 ms
     MPLS Label=100416 CoS=0 TTL=1 S=1
 3  192.168.32.1 (192.168.32.1)  1.377 ms  0.979 ms  0.921 ms
```

Providing LSP Access to Just the Ingress Router

The default action of placing the egress router address in the `inet.3` routing table can be altered within the JUNOS software in multiple ways. Using various options within the `traffic-engineering` command, you can move the contents of the `inet.3` table into `inet.0`, or you can copy the contents into `inet.0`. To complicate matters further, the copy functionality can have multiple effects on the router itself. Let's discuss how each of the functions works and provide some rationale for using each.

Using *traffic-engineering bgp-igp*

When you use the `bgp-igp` option within your MPLS configuration, the router moves the contents of the `inet.3` table into `inet.0`. The movement of these addresses doesn't alter the default BGP recursive lookup functionality as the protocol inspects both the `inet.3` and `inet.0` tables. What you gain as a benefit, however, is the ability for the ingress router to locate the LSP during a normal IP routing lookup.

The Sherry router in Figure 8.10 has established the ***Sherry-to-Char*** LSP in the network. We see that the default action of placing the egress address in the `inet.3` routing table has occurred:

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

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:05, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
```

From the perspective of the ingress router, however, all IP packets with a destination address of 192.168.32.1 use the route installed by IS-IS. These packets do not use the LSP for forwarding across the network since all routing lookups are performed in the `inet.0` routing table:

```
user@Sherry> show route 192.168.32.1

inet.0: 29 destinations, 29 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[IS-IS/18] 00:04:35, metric 30
                    > to 10.222.29.2 via ge-0/2/0.0
                      to 10.222.28.2 via at-0/1/0.0

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:01:15, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
```

Sherry applies the `bgp-igp` knob to its MPLS configuration to move the `inet.3` routes into the `inet.0` routing table:

```
user@Sherry> show configuration protocols mpls
traffic-engineering bgp-igp;
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    primary via-Sangiovese;
}
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;

user@Sherry> show route 192.168.32.1

inet.0: 29 destinations, 30 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:46, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
                    [IS-IS/18] 00:00:50, metric 30
                    > to 10.222.29.2 via ge-0/2/0.0
                      to 10.222.28.2 via at-0/1/0.0
```

Sherry now has the ability to use the LSP for traffic forwarding for packets address to the 192.168.32.1 address:

```
user@Sherry> traceroute 192.168.32.1
traceroute to 192.168.32.1 (192.168.32.1), 30 hops max, 40 byte packets
 1  10.222.28.2 (10.222.28.2)  1.425 ms  1.215 ms  1.163 ms
     MPLS Label=100448 CoS=0 TTL=1 S=1
 2  10.222.4.2 (10.222.4.2)  1.293 ms  0.924 ms  1.126 ms
     MPLS Label=100448 CoS=0 TTL=1 S=1
 3  192.168.32.1 (192.168.32.1)  0.836 ms  0.957 ms  1.169 ms
```

Using *traffic-engineering bgp-igp-both-ribs*

While some administrators find it advantageous to place the LSP egress address in the `inet.0` table, one possible downside does exist. Specifically, the use of the MPLS network to support virtual private networks (VPN) requires the egress address to be located in the `inet.3` routing table. To resolve this issue, the JUNOS software provides the `bgp-igp-both-ribs` option for the `traffic-engineering` command. This command copies the egress addresses into the `inet.0` routing table instead of moving them.

The Sherry router is operating in a default mode for the 192.168.32.1 egress address of the ***Sherry-to-Char*** LSP:

```
user@Sherry> show route 192.168.32.1

inet.0: 29 destinations, 29 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[IS-IS/18] 00:00:04, metric 30
                    > to 10.222.29.2 via ge-0/2/0.0
                      to 10.222.28.2 via at-0/1/0.0

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:04, metric 65535
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
```

The configuration of the `traffic-engineering bgp-igp-both-ribs` command on Sherry copies the address into the `inet.0` routing table:

```
user@Sherry> show configuration protocols mpls
traffic-engineering bgp-igp-both-ribs;
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    primary via-Sangiovese;
}
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;

user@Sherry> show route 192.168.32.1

inet.0: 29 destinations, 30 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:41, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
                    [IS-IS/18] 00:00:46, metric 30
                      to 10.222.29.2 via ge-0/2/0.0
                    > to 10.222.28.2 via at-0/1/0.0

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:41, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
```

As before, IP packets with a destination address of 192.168.32.1 are forwarded by the Sherry router across the LSP:

```
user@Sherry> traceroute 192.168.32.1
traceroute to 192.168.32.1 (192.168.32.1), 30 hops max, 40 byte packets
 1  10.222.28.2 (10.222.28.2)  1.303 ms  1.393 ms  1.071 ms
     MPLS Label=100448 CoS=0 TTL=1 S=1
 2  10.222.4.2 (10.222.4.2)  1.310 ms  1.465 ms  1.146 ms
     MPLS Label=100448 CoS=0 TTL=1 S=1
 3  192.168.32.1 (192.168.32.1)  0.828 ms  0.966 ms  1.160 ms
```

Using *traffic-engineering mpls-forwarding*

The main disadvantage to using the `bgp-igp-both-ribs` command is how the router views the LSP addresses in the `inet.0` routing table. By default, these routes have a better (lower) JUNOS software protocol preference value than the version of the route placed there by the IGP. This makes the LSP version of the address active in the routing table. As only active routes are eligible for routing policy processing, the placement of the LSP addresses might cause existing routing policies to not perform as expected. This is especially true when the routing policy is exporting the IGP routes into another protocol, like BGP.

To combat this issue, you can configure MPLS with the `traffic-engineering mpls-forwarding` command. This allows the LSP egress addresses to be copied to the `inet.0` routing table, but only marks them as active for purposes of forwarding packets. The IGP versions of these same addresses are left active for routing purposes. In effect, this command provides the best of all possible situations.

The Sherry router is currently configured with this command. From the router's output we can see that the different versions of the 192.168.32.1 route are notated differently in the routing table:

```
user@Sherry> show configuration protocols mpls
traffic-engineering mpls-forwarding;
label-switched-path Sherry-to-Char {
    to 192.168.32.1;
    primary via-Sangiovese;
}
path via-Sangiovese {
    192.168.24.1 loose;
}
interface all;

user@Sherry> show route 192.168.32.1

inet.0: 29 destinations, 30 routes (25 active, 0 holddown, 4 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    @[IS-IS/18] 00:00:08, metric 30
                      to 10.222.29.2 via ge-0/2/0.0
                    > to 10.222.28.2 via at-0/1/0.0
                   #[RSVP/7] 00:00:04, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

192.168.32.1/32    *[RSVP/7] 00:00:04, metric 30
                    > via at-0/1/0.0, label-switched-path Sherry-to-Char
```

As we would expect, IP packets use the LSP for forwarding on the Sherry router:

```
lab@Sherry> traceroute 192.168.32.1
```

```
traceroute to 192.168.32.1 (192.168.32.1), 30 hops max, 40 byte packets
 1  10.222.28.2 (10.222.28.2)  1.531 ms  1.345 ms  1.109 ms
     MPLS Label=100464 CoS=0 TTL=1 S=1
 2  10.222.4.2 (10.222.4.2)  1.369 ms  0.895 ms  1.152 ms
     MPLS Label=100464 CoS=0 TTL=1 S=1
 3  192.168.32.1 (192.168.32.1)  0.901 ms  0.915 ms  1.160 ms
```

Summary

In this chapter, we examined how the JUNOS software constructs a traffic engineering database (TED) in the network. Information specific to MPLS, such as current reservations and available bandwidth, is propagated within the interior gateway protocols. OSPF uses a Type 10 Opaque LSA for this purpose, whereas IS-IS uses the Extended IS Reachability TLV, number 22. We discussed the steps of the Constrained Shortest Path First (CSPF) algorithm, which evaluates the information in the TED to compute a path from the ingress router to the egress router. User-defined constraints, including administrative groups and priority values, can be applied to the CSPF algorithm, and we discussed those next.

We then explored some methods available within the JUNOS software for protecting LSP traffic flows in the network. This discussion centered on the creation of primary and secondary paths for a single LSP. We further found that the secondary path could be placed into a hot-standby mode to quickly move traffic in the event of a failure. We examined the operation of fast reroute to further protect user traffic already contained within the LSP. We saw that both node protection and link protection created alternate paths through the network to quickly alter the traffic flows on a transit router should a failure occur.

We concluded the chapter with a discussion of various ways to control the behavior of the LSPs in your network. We first talked about adaptive mode, which establishes bandwidth reservations using the shared explicit style. This allows multiple paths within the main RSVP session to share the single reservation instead of double-counting the bandwidth. In addition, adaptive mode allows the ingress router to create a new LSP before tearing down the existing LSP in a make-before-break mode. After a quick look at having the egress router advertise a label value of 0, we saw two methods for hiding the topology of the LSP. We accomplished this by changing the ingress and penultimate router's default behavior with regard to the time-to-live value in the MPLS header. We then discussed methods for allowing other protocols to access the LSP for traffic forwarding. The first of these was the advertisement of the LSP as a point-to-point link within the IGPs themselves. The second was the use of the `traffic-engineering` command on the ingress router to allow IP route lookups to access the LSP.

Exam Essentials

Be able to describe how information is placed into the traffic engineering database. Both OSPF and IS-IS have been extended to carry traffic engineering specific information within their protocol updates. The Type 10 Opaque LSA and the Extended IS Reachability TLV contain bandwidth, administrative group, and addressing information. These advertisements are then placed into the TED on each IGP router.

Know the steps used by the CSPF algorithm to locate a path for the LSP. The algorithm first removes all links from the TED that don't meet the bandwidth requirements of the LSP. Then all links that do not contain any included administrative group information are removed. If the LSP is configured to exclude any administrative group information, those links are then removed by the algorithm. Using the remaining links, the algorithm locates the shortest path through the network from the ingress to the egress router.

Be familiar with methods available to the ingress router for protecting MPLS traffic flows. An individual LSP may be configured for both a primary and a secondary path through the network. When the primary path is available, it is used for traffic forwarding. In a failure mode, the ingress router establishes the secondary path and begins forwarding traffic using MPLS labels once again. The failover time on the ingress router can be shortened by allowing the secondary path to be established in the network prior to the primary path failure. This is accomplished through the use of the `standby` command.

Be able to describe how user traffic within an LSP is protected from failures. Once the ingress router has begun forwarding traffic along the LSP, a network failure in the path can cause those packets to be dropped from the network. To alleviate this problem, the LSP can be configured for either node or link protection fast reroute. These methods allow all routers in the LSP to preestablish paths in the network to avoid the next downstream link or node. During a failure mode, the device that notices the failure immediately begins using the temporary paths to continue forwarding traffic within the LSP.

Be able to describe how multiple LSP paths share bandwidth reservations. By default, all newly established paths in the network reserve unique and distinct bandwidth reservations using the fixed filter style. Within a single RSVP session, multiple paths can share a reservation when the shared explicit style is used during the path establishment. Within the JUNOS software, the `adaptive` command accomplishes this goal. In addition, adaptive mode allows the ingress router to use a make-before-break system for rerouting an LSP in the network. The new path is established and traffic is moved before the old path is torn down.

Understand methods used to provide LSP visibility to non-BGP learned routes. Advertising the LSP to the IGP link-state database is one effective method for accomplishing this goal. Each router in the network then sees the LSP as a point-to-point link with some metric value attached. This information is used in the SPF calculation on the router, which may result in the LSP being used for traffic forwarding.

Review Questions

1. How is traffic engineering information propagated in an OSPF network?
   - **A.** Type 8 Opaque LSA
   - **B.** Type 9 Opaque LSA
   - **C.** Type 10 Opaque LSA
   - **D.** Type 11 Opaque LSA
2. How is traffic engineering information propagated in an IS-IS network?
   - **A.** IS Reachability TLV
   - **B.** Internal IP Reachability TLV
   - **C.** Extended IS Reachability TLV
   - **D.** Extended IP Reachability TLV
3. Which CSPF constraint is carried in a 32-bit vector?
   - **A.** Bandwidth
   - **B.** Administrative groups
   - **C.** Named path ERO
   - **D.** Load-balancing configuration
4. What is the default JUNOS software setting for RSVP priority values?
   - **A.** A setup of 0 and a hold of 7
   - **B.** A setup of 7 and a hold of 0
   - **C.** A setup of 0 and a hold of 0
   - **D.** A setup of 7 and a hold of 7
5. What is the maximum number of primary paths allowed on an LSP?
   - **A.** 0
   - **B.** 1
   - **C.** 2
   - **D.** 3
6. What JUNOS software command allows an LSP path to be established before it is needed for a failure condition?
   - **A.** `admin-group`
   - **B.** `optimize` ***`timer`***
   - **C.** `standby`
   - **D.** `traffic-engineering`

7. What fast reroute mode permits each router in the path to create a detour path to the egress router?
   A. Node protection
   B. Link protection
   C. Detour protection
   D. Bypass protection

8. What fast reroute mode permits each router in the path to create a bypass path to the next downstream router?
   A. Node protection
   B. Link protection
   C. Detour protection
   D. Bypass protection

9. What RSVP reservation style results from configuring `adaptive` on an LSP?
   A. Fixed filter
   B. Wildcard filter
   C. Shared explicit
   D. Wildcard explicit

10. Which JUNOS software configuration command copies entries from the `inet.3` routing table to the `inet.0` routing table while leaving the previous `inet.0` entries active for routing purposes?
    A. `traffic-engineering`
    B. `traffic-engineering bgp-igp`
    C. `traffic-engineering bgp-igp-both-ribs`
    D. `traffic-engineering mpls-forwarding`

Answers to Review Questions

1. C. OSPF uses the Type 10 Opaque LSA to propagate traffic engineering information to the network. This LSA has an area flooding scope that results in each area having a separate and distinct TED.
2. C. The Extended IS Reachability TLV is used by IS-IS to propagate traffic engineering information to the network. This TLV is flooded within a specific level that results in each level having a separate and distinct TED.
3. B. MPLS administrative groups are each assigned a value between 0 and 31. These values are each represented by a bit in a 32-bit vector propagated by the IGPs.
4. B. By default, each LSP is established with a setup priority value of 7 and a hold priority value of 0.
5. B. While an LSP can be configured without a primary path, it can contain only one. When the constraints of this path are met, it is used by the LSP for forwarding traffic.
6. C. The `standby` command is configured on a secondary path to establish the path in the network before the primary path fails.
7. A. The node-protection version of fast reroute allows each LSP router to create and establish a detour path to the egress router. During a failure mode, the point of local repair swaps the incoming label for the detour label and forwards the packet.
8. B. The link-protection version of fast reroute allows each LSP router to create and establish a bypass path to the next downstream router. During a failure mode, the point of local repair swaps the incoming label for the label advertised by the downstream router and pushes the bypass label onto the stack.
9. C. The shared explicit (SE) reservation style is used when an LSP is configured with the `adaptive` command.
10. D. All route entries in the `inet.3` routing table are copied to `inet.0` when the MPLS process is configured with the `traffic-engineering mpls-forwarding` command. Additionally, the existing `inet.0` entries are marked for routing use only while the new entries are marked for forwarding use only.

Chapter 9

Layer 2 and Layer 3 Virtual Private Networks

JNCIS EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ **Identify the differences between a CPE and a provider-provisioned VPN**
- ✓ **Describe the operation of the control plane in a Layer 3 VPN environment**
- ✓ **Describe the operation of the data plane in a Layer 3 VPN environment**
- ✓ **Identify the steps involved in configuring a Layer 3 VPN**
- ✓ **Identify the differences between a Layer 2 and a Layer 3 VPN**
- ✓ **Describe the steps involved in configuring a Kompella Layer 2 VPN**
- ✓ **Describe the steps involved in configuring a Martini Layer 2 VPN**

In this chapter, we begin our discussion of virtual private networks (VPNs) by discussing the terminology used within the JUNOS software for configuring and deploying a VPN. We then focus our attention on the type of VPN that interacts with the customer's routing domain using Layer 3 routing protocols. We see how the control plane and the data plane of the VPN work within this constraint. After configuring a basic Layer 3 VPN using different routing protocols for the customer communications, we examine methods for providing Internet access to customer sites.

We then continue our examination of VPNs by focusing our attention on Layer 2 protocols transported across a provider network. After discussing the two types of Layer 2 VPNs available within the JUNOS software, we examine each of the varieties in further detail. Within the context of each method, we discuss the advertisement of Layer 2 connection information between the provider edge (PE) routers. We then see how the actual data is transmitted across the provider network using each of the Layer 2 VPN methods.

From a conceptual standpoint, a VPN is quite simple. It's the operation of a network over a common physical infrastructure. The provider of the infrastructure allows each customer to remain segregated within their own virtual connections. In fact, we've been running VPNs to interconnect locations for quite some time now. Three examples of these "traditional" VPNs include leased lines, Frame Relay, and Asynchronous Transfer Mode (ATM) circuits. In today's nomenclature, however, this type of connectivity is usually not associated with VPNs. This mainly arises from the fact that all access to the Internet, as well as remote locations, is driven from a connection to a provider that often uses Frame Relay or ATM for access. As such, we normally consider a VPN to connect multiple locations together using an IP-based technology. In fact, we're going to make that assumption for our discussion and focus on the methods used for connecting those locations.

VPN Basics

Before discussing either Layer 3 or Layer 2 VPNs, we first need to examine some terms that relate to both technologies as well as some terms that are specific to one or the other. Unfortunately for us, the use of a VPN within a provider network is filled with multiple terms and acronyms. So, let's see what they all mean.

Customer edge router The *customer edge* (CE) router is the device physically located in the customer's location. It connects to the provider network using a Layer 2 technology such as

Frame Relay, ATM, or Ethernet. Within a Layer 3 (L3) VPN, the CE router forms a protocol relationship with the provider router to exchange routing knowledge. The CE router may be the only device in a particular customer location, or it might participate in a routing domain based in that location. For a Layer 2 VPN, the CE router transmits frames to the provider network that are destined for a remote customer location. The CE router may have a single static route that points at the provider, or it may be communicating via a routing protocol with other CE routers in remote locations.

Provider edge router The *provider edge* (PE) router is located in the provider's network and communicates directly with the CE router. In addition, it maintains relationships with other routers inside the provider's network. For an L3VPN environment, the PE router communicates with its attached CE router to receive routing updates. This information is then advertised to a remote PE router that is connected to another of the customer's sites. When the PE router receives data packets destined for a remote site, it forwards the packets using a Multiprotocol Label Switching (MPLS) LSP across the provider's network. When the PE router is participating in a Layer 2 VPN, it simply receives a Layer 2 frame from the local CE router, which it forwards to a remote PE router using an MPLS LSP.

Provider router The *provider* (P) router is located within the core of the provider's network. The P routers do not maintain any knowledge of the customer's VPN information but simply forward MPLS packets from one PE router to another.

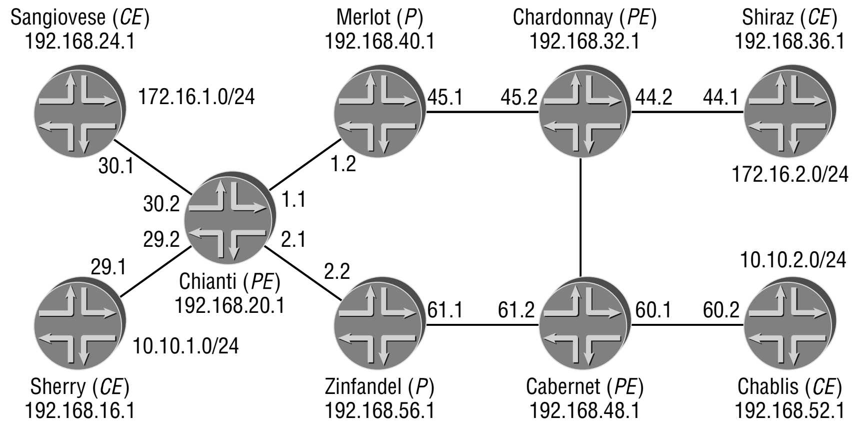
VPN routing and forwarding table In a Layer 3 VPN, each PE router maintains a separate *VPN routing and forwarding* (VRF) table for each customer connected to that local PE. Each VRF table contains routing knowledge specific to the customer's network. When the attached CE router advertises routes from that location, the PE router places them into that customer's VRF table. The routes are then advertised to the remote PE routers, which also service this particular customer. Each remote PE router places the received routes in that customer's VRF table and advertises them to their locally attached CE routers.

VPN forwarding table A *VPN forwarding table* (VFT) is used in a Layer 2 VPN environment. Each PE router creates a separate VFT for each customer connected to that PE router. The VFT contains information that describes the local PE-CE connection, such as encapsulation type, local logical interface, a local site identifier, and some MPLS label information. Each VFT contains knowledge of the remote locations connected across the provider's LSPs.

VPN connection table The information contained in a Layer 2 VFT is transmitted to the remote PE routers in a *VPN connection table* (VCT). Each VCT is a subset of the information contained in the VFT for that particular customer. This allows each PE to forward the received Layer 2 frames to the appropriate remote PE router.

Figure 9.1 displays a typical provider network that is supporting VPNs. The Sangiovese, Sherry, Shiraz, and Chablis routers are all CE routers, whereas the Chianti, Chardonnay, and Cabernet routers are PE routers. The only P routers in the network are Merlot and Zinfandel.

FIGURE 9.1 Provider-provisioned VPN environment



Layer 3 VPNs

Within a Layer 3 VPN, the customer advertises IP routing knowledge to the provider network. The ISP then advertises this routing information across its network to the other customer locations. This simple concept requires quite a bit of coordination between the provider and the customer. In addition, the provider network must be configured to support the advertisement of these routes. To accomplish this, we first need a method for ensuring that the customer routes remain segregated. This is the function of specially designed *Network Layer Reachability Information* (NLRI), which we discuss next.

VPN Network Layer Reachability Information

One of the core principles of operating a VPN is maintaining separation between the customer networks. The normal rules of Border Gateway Protocol (BGP) routing make this a difficult task since multiple versions of the same route are parsed through the route selection algorithm and a single best path is selected. It is only this single best path that is readvertised to all other BGP peers.

To better understand why this is an issue, let's assume that a provider is offering VPN services to both Customer A and Customer B. Each of the customers is using the Request for Comments (RFC) 1918 address space of 192.168.0.0 /16 within their network. When either customer sends routes to the local PE router, the routes are placed into the VRF table associated with that customer. In our example, the 192.168.1.0 /24 route arrives from Customer A on Router A within the provider's network while the same 192.168.1.0 /24 route from Customer B arrives on Router B. Both Router A and Router B advertise the 192.168.1.0 /24 route to Router C in the provider's core. From Router C's perspective, it receives the same route with different attributes from different peers. It then runs the route selection algorithm and selects the route from Router A as the local active route. This is readvertised to Router D on the far edge of the network, where both Customer A and Customer B have VPN connections. Router D sends the route to both of the customer routers, which install it in their local routing tables. The main problem, of course, is that when devices in Customer B's network send packets destined for the 192.168.1.0 /24 network, they are all delivered to Customer A. Since this version of the route was selected by BGP, the physical next hops for that destination were also installed in the routing table. This means that all traffic flows to that same destination. Clearly, this scenario doesn't work for a VPN environment.

To alleviate the issue we just outlined, a Layer 3 VPN uses a special format for representing customer routes within the provider's network. This format allows each provider router to uniquely view routes from different customers as separate, even when they advertise the same IPv4 prefix. Figure 9.2 displays this format, which includes the following fields:

Mask (1 octet) This field displays the total length of the NLRI advertised within the BGP Update message. The value of the mask varies from 88 to 120.

MPLS Label (3 octets) Each route advertised in a Layer 3 VPN is associated with an MPLS label, which is contained in this field. This label value is allocated by the PE router which first advertises the route into the provider's network. It is used for forwarding data packets to the appropriate CE router.

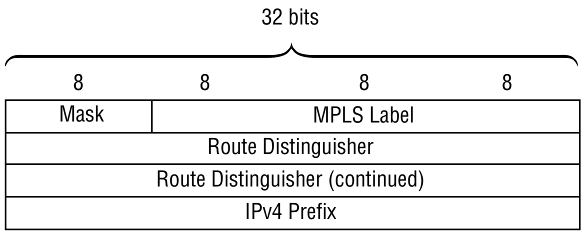
Route Distinguisher (8 octets) The Route Distinguisher field is critical to the operation of a Layer 3 VPN. Within this field lies a specific value unique to each customer VPN or customer site. It is the route distinguisher (RD) that allows each provider router to view the customer IP addresses as separate and unique entities.

IPv4 Prefix (Variable) This field displays the customer route being advertised between the PE routers.

Of course, the advertisement of specialized NLRI requires the establishment of multiprotocol BGP (MBGP) sessions between the PE routers. These sessions are established between individual peers based on the settings within the `family` command at the global, group, or neighbor level. (We discuss MBGP in Chapter 5, "Advanced Border Gateway Protocol (BGP)".) The PE routers of Chianti, Chardonnay, and Cabernet are configured to advertise the VPN-IPv4 NLRI to each other. For example, the configuration of Chianti is as follows:

```
user@Chianti> show configuration protocols bgp
group Internal-Peers {
    type internal;
    local-address 192.168.20.1;
    family inet-vpn {
        unicast;
    }
    neighbor 192.168.32.1;
    neighbor 192.168.48.1;
}
```

FIGURE 9.2 The VPN-IPv4 NLRI format



Once the sessions are established, the received NLRI appears in the bgp.l3vpn.0 routing table:

```
user@Chianti> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0            5          5          0          0          0          0
Peer               AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State
192.168.32.1    65432      13974      14002       0       4 3d 19:56:13 Establ
  bgp.l3vpn.0: 2/2/0
  VPN-A.inet.0: 2/2/0
192.168.48.1    65432      13984      14019       0       0 4d 19:57:37 Establ
  bgp.l3vpn.0: 3/3/0
  VPN-B.inet.0: 3/3/0
```

Suppose the Chianti and Chardonnay routers are communicating routes for a VPN customer connected to each of them. On Chardonnay the VPN-IPv4 routing information appears within the bgp.l3vpn.0 table:

```
user@Chardonnay> show route table bgp.l3vpn.0

bgp.l3vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.20.1:3:10.222.30.0/24
                   *[BGP/170] 01:24:05, localpref 100, from 192.168.20.1
                      AS path: I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti
```

The customer prefix being announced by Chianti is 10.222.30.0 /24. The leading information in the router output (192.168.20.1:3) is the RD appended by Chianti to provide unique routing information. Let's discuss that next.

Route Distinguishers

The *Route Distinguisher* is an 8-octet field that actually consists of three portions; the Type, the Administrator, and Assigned Number fields. The first 2 octets of the Route Distinguisher contain the type of RD being used, which determines the length of the remaining fields. A type code of 0 represents an Administrator field that is 2 octets long, whereas the Assigned Number field is 4 octets in length. A type code of 1, on the other hand, means that the Administrator field is 4 octets long and the Assigned Number field is 2 octets long.

FIGURE 9.3 Route distinguisher formats

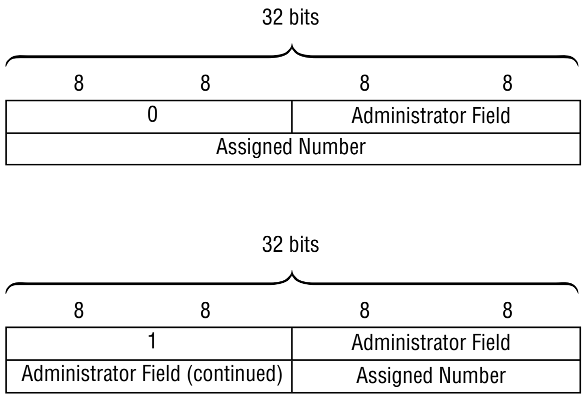


Figure 9.3 displays the two possible formats for RDs. The actual RD used within a Layer 3 VPN is entirely up to the discretion of the VPN's network administrators. The only requirement for assigned RD values is to keep them unique throughout the network, on either a per-VRF basis or a per-VPN basis. There are, however, some common best practices in use today. When the RD is using a 2-octet Administrator field, place the global Autonomous System value in this field. The router ID of the originating PE router should be used when the Administrator field is 4 octets long. In either case, the value placed in the Assigned Number field is left to the discretion of the administrator.

Within the JUNOS software, there are two main methods for assigning an RD to a particular VRF table—automatically or manually. When you use the `route-distinguisher-id` command within the `[edit routing-options]` configuration hierarchy, the PE automatically generates an RD for each unique VRF table enabled on that router. This process uses the router ID of the advertising PE router within the Administrator field and generates an assigned number automatically. As an example, the Chianti router in Figure 9.1 is currently advertising routes for two customers. We've entered Chianti's router ID within the `route-distinguisher-id` command:

```
user@Chianti> show configuration routing-options
route-distinguisher-id 192.168.20.1;
autonomous-system 65432;
```

This allows Chianti to advertise routes from each customer with a different RD to its established PE peers. We see one route appear on Chardonnay, whereas another arrives at the Cabernet router:

```
user@Chardonnay> show route table bgp.l3vpn.0

bgp.l3vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.20.1:3:10.222.30.0/24
                   *[BGP/170] 01:54:16, localpref 100, from 192.168.20.1
                      AS path: I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti

user@Cabernet> show route table bgp.l3vpn.0

bgp.l3vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.20.1:4:10.222.29.0/24
                   *[BGP/170] 00:07:44, localpref 100, from 192.168.20.1
                      AS path: I
                    > via so-0/1/1.0, label-switched-path Cab-to-Chianti
```

Although both values begin with the router ID of 192.168.20.1, the different Assigned Number fields provide a unique RD for each customer VRF table.

Of course, the JUNOS software also provides you the ability to manually control the RD value. Some administrators prefer to use an AS-based RD, whereas others would like more control over the value placed in the Assigned Number field. Either way, you configure a manual RD value within the `[edit routing-instances]` configuration hierarchy corresponding to the VRF table of a particular customer. For example, the Cabernet router assigns a manual RD as so:

```
user@Cabernet> show configuration routing-instances
VPN-B {
    instance-type vrf;
    interface so-0/1/2.0;
    route-distinguisher 65432:8989;
    vrf-target target:65432:2222;
}
```

This value appears within the `bgp.l3vpn.0` routing table on the Chianti router:

```
user@Chianti> show route table bgp.l3vpn.0

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
```

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

192.168.32.1:3:10.222.44.0/24
                   *[BGP/170] 00:50:48, localpref 100, from 192.168.32.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
65432:8989:10.222.60.0/24
                   *[BGP/170] 00:50:48, localpref 100, from 192.168.48.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Cab
```

NOTE

Other important commands to use in the configuration of a Layer 3 VPN include `instance-type`, `interface`, and `protocols`. The `instance-type` command informs the router what type of VPN service to operate for the customer while the `interface` command places the customer's interface into the VRF table. Finally, the specific routing protocol operating between the PE and CE routers is configured within the `protocols` section of the routing instance.

Basic Operational Concepts

When you are using a Layer 3 VPN, you need to be aware of the operation of both the control plane (route advertisements) and the forwarding plane (packet forwarding). We examine both topics within this section; let's begin with route advertisements. After all, you can't forward packets when you don't have a route for the destination.

The Control Plane

Customer routes in a Layer 3 VPN environment are advertised between PE routers within the provider's network. To ensure that the routes are delivered to each appropriate customer site, the provider has two main choices. The first option involves the PE router forming only a protocol relationship, BGP in our case, with the other PE routers connected to that particular customer. This approach has the main disadvantage of requiring a new BGP connection for each new customer site added to a different PE router. The second option uses a full mesh of BGP connections between the PE routers and routing policy to control which routes are accepted within each customer VRF table. Because this option carries with it the ability to easily change the logical topology, many real-world networks adopt this model. Therefore, we'll focus our attention in this direction.

The routing policies used by the PE routers in a Layer 3 VPN rely on a *route target* to identify routes belonging to a specific VRF table. The route target takes the form of a BGP extended community and is assigned to all routes advertised out of a particular VRF table. (We discuss extended communities in Chapter 4, "Border Gateway Protocol (BGP).") When the receiving PE router sees the route target associated with a local VRF table, it accepts the route advertisement and places the route into the customer's VRF table. Using the routers in Figure 9.1, let's examine each step in the process:

1. The Chardonnay router receives routes in the customer's VRF table that belong to that particular customer. In our sample network, these are direct and static routes but may also include routes received from a routing protocol operating between the PE and CE routers. The JUNOS software currently supports BGP, OSPF, and RIP for use as CE-PE routing protocols.

```
user@Chardonnay> show route table VPN-A.inet.0

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

10.222.44.0/24     *[Direct/0] 2d 00:49:16
                    > via so-0/1/0.0
10.222.44.2/32     *[Local/0] 2d 00:49:18
                      Local via so-0/1/0.0
172.16.2.0/24      *[Static/5] 1d 20:00:29
                    > to 10.222.44.1 via so-0/1/0.0
```

2. Chardonnay then consults the export policy applied to this VRF table to determine which routes should be advertised to the other PE routers in the network. This policy also applies the appropriate route target to the advertised routes. Because Chardonnay is configured with the `vrf-target` command, it automatically advertises all active routes that belong to this particular VRF table. In addition, the route target of `target:65432:1111` is applied to the advertised routes:

```
user@Chardonnay> show configuration routing-instances
VPN-A {
    instance-type vrf;
    interface so-0/1/0.0;
    vrf-target target:65432:1111;
    routing-options {
        static {
            route 172.16.2.0/24 next-hop 10.222.44.1;
        }
    }
}
```

Using this configuration, Chardonnay advertises the direct and static routes within the VRF table to Chianti:

```
user@Chardonnay> show route advertising-protocol bgp 192.168.20.1 detail

VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 10.222.44.0/24 (1 entry, 1 announced)
```

```
 BGP group int type Internal
     Route Distinguisher: 192.168.32.1:3
     VPN Label: 100128
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111

* 172.16.2.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 192.168.32.1:3
     VPN Label: 100144
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111
```

NOTE

You can manually apply an export policy to the VRF table with the `vrf-export` command. Use this command when a subset of routes should be advertised to remote PE routers.

3. The receiving PE router, Chianti in our example, examines the incoming routes and compares them to the import policy applied to the VRF table. This policy matches on all routes that contain the route target assigned to the customer's VRF table. Chianti is configured in a similar manner to Chardonnay in that it uses the `vrf-target` command to assign a route target of `target:65432:1111` to the VRF table:

```
user@Chianti> show configuration routing-instances
VPN-A {
    instance-type vrf;
    interface so-0/1/2.0;
    vrf-target target:65432:1111;
    routing-options {
        static {
            route 172.16.1.0/24 next-hop 10.222.30.1;
        }
    }
}
```

Routes that match the import policy are placed into the `bgp.l3vpn.0` routing table on the local PE router:

```
user@Chianti> show route table bgp.l3vpn.0
```

```
bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.32.1:3:10.222.44.0/24
                   *[BGP/170] 1d 20:23:19, localpref 100, from 192.168.32.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
192.168.32.1:3:172.16.2.0/24
                   *[BGP/170] 1d 20:22:54, localpref 100, from 192.168.32.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
```

NOTE

By default, routes that don't match a local route target are not installed in the Adjacency-RIB-In table by the JUNOS software.

NOTE

You may also manually apply an import policy to a VRF table using the `vrf-import` command. Use this command when you want to accept only a subset of routes from remote PE routers.

4. The receiving PE router examines the received routes to determine the address listed in the BGP Next Hop attribute. For the routes advertised by Chardonnay, this value is the local BGP peering address of 192.168.32.1 (`Self`):

```
user@Chardonnay> show route advertising-protocol bgp 192.168.20.1 detail

VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 10.222.44.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 192.168.32.1:3
     VPN Label: 100128
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111

* 172.16.2.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 192.168.32.1:3
     VPN Label: 100144
```

```
    Nexthop: Self
    Localpref: 100
    AS path: I
    Communities: target:65432:1111
```

Chianti, the receiving router, then determines if the BGP Next Hop is available in the `inet.3` routing table. This ensures that an MPLS LSP is established from the local PE router to the remote PE router for forwarding user traffic within the VPN:

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

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

192.168.32.1/32    *[RSVP/7] 2d 23:15:46, metric 20
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
192.168.48.1/32    *[RSVP/7] 2d 23:15:46, metric 20
                    > via so-0/1/0.0, label-switched-path Chianti-to-Cab
```

5. The label value used to reach the remote PE router is associated with the routes received from that router. For the ***Chianti-to-Char*** LSP, Chianti pushes a label value of 100,448:

```
user@Chianti> show route 192.168.32.1 table inet.3 detail

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
192.168.32.1/32 (1 entry, 1 announced)
        State: <FlashAll>
        *RSVP   Preference: 7
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Char
                Label operation: Push 100448
                State: <Active Int>
                Local AS: 65432
                Age: 2d 23:19:27        Metric: 20
                Task: RSVP
                Announcement bits (1): 0-Resolve inet.3
                AS path: I
```

The MPLS label advertised with the route, the `VPN Label`, is also associated with the received routes in the `bgp.l3vpn.0` table. It is placed in a label stack along with the label used to reach the remote PE router:

```
user@Chianti> show route table bgp.l3vpn.0 detail

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
```

```
192.168.32.1:3:10.222.44.0/24 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.32.1:3
                Source: 192.168.32.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Char
                Label operation: Push 100128, Push 100448(top)
                Protocol next hop: 192.168.32.1
                Push 100128
                 Indirect next hop: 8522000 325
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 1d 20:39:55        Metric2: 20
                Task: BGP_65432.192.168.32.1+179
                AS path: I
                Communities: target:65432:1111
                VPN Label: 100128
                Localpref: 100
                Router ID: 192.168.32.1
                Secondary Tables: VPN-A.inet.0

192.168.32.1:3:172.16.2.0/24 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.32.1:3
                Source: 192.168.32.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Char
                Label operation: Push 100144, Push 100448(top)
                Protocol next hop: 192.168.32.1
                Push 100144
                 Indirect next hop: 8522150 362
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 1d 20:39:30        Metric2: 20
                Task: BGP_65432.192.168.32.1+179
                AS path: I
                Communities: target:65432:1111
                VPN Label: 100144
                Localpref: 100
                Router ID: 192.168.32.1
                Secondary Tables: VPN-A.inet.0
```

6. The received routes are placed into the VRF table corresponding to the appropriate customer. They are installed as BGP routes since they were received via that protocol:

   ```
   user@Chianti> show route table VPN-A protocol bgp

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

   10.222.44.0/24     *[BGP/170] 1d 20:42:18, localpref 100, from 192.168.32.1
                         AS path: I
                       > via so-0/1/0.0, label-switched-path Chianti-to-Char
   172.16.2.0/24      *[BGP/170] 1d 20:41:53, localpref 100, from 192.168.32.1
                         AS path: I
                       > via so-0/1/0.0, label-switched-path Chianti-to-Char
   ```

7. The PE router forwards the received routes to the attached CE router using the routing protocol operating between the two routers. In our case, Chianti and Sangiovese are using static routes for reachability, so no protocol advertisements are necessary.

The Data Forwarding Plane

Once the appropriate routes are advertised and received by the CE routers, the customer can begin forwarding data packets through the provider network. Suppose that the Sangiovese router in Figure 9.1 sends an ICMP echo request packet to the 172.16.2.1 address. The forwarding of the packet follows these basic steps:

1. The CE router of Sangiovese performs a route table lookup to locate a route to 172.16.2.1. It finds a locally configured static route for that destination with a next-hop value of 10.222.30.2. It then forwards the packet to Chianti across its `so-0/2/0.0` interface:

   ```
   user@Sangiovese> show route 172.16.2.1

   inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
   + = Active Route, - = Last Active, * = Both

   172.16.2.0/24      *[Static/5] 1d 20:47:47
                       > to 10.222.30.2 via so-0/2/0.0
   ```

2. Chianti receives the packet on the interface configured within the VRF table associated with the CE router. It performs a route table lookup within that VRF table to locate a route to 172.16.2.1:

   ```
   user@Chianti> show route 172.16.2.1 table VPN-A

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

```
172.16.2.0/24      *[BGP/170] 1d 20:50:59, localpref 100, from 192.168.32.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
```

The result tells Chianti to forward the packet out its `so-0/1/0.0` interface along the ***Chianti-to-Char*** LSP. A more detailed examination of the routing table contents reveals important information concerning this forwarding action:

```
user@Chianti> show route 172.16.2.1 table VPN-A detail

VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
172.16.2.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.32.1:3
                Source: 192.168.32.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Char
                Label operation: Push 100144, Push 100448(top)
                Protocol next hop: 192.168.32.1
                Push 100144
                 Indirect next hop: 8522150 362
                State: <Secondary Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 1d 20:53:20        Metric2: 20
                Task: BGP_65432.192.168.32.1+179
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: target:65432:1111
                VPN Label: 100144
                Localpref: 100
                Router ID: 192.168.32.1
                Primary Routing Table bgp.l3vpn.0
```

Chianti actually performs a double push operation on the forwarded packet. The VPN label of 100,144 is placed on the bottom of the stack, and the 100,448 label is placed on the top of the stack.

3. The Merlot router, a P router, receives an MPLS packet from Chianti. It examines the top label value of 100,448 and performs a switching table lookup for that label value. The result tells Merlot to pop the top MPLS label and forward the remaining data out its `so-0/1/2.0` interface to Chardonnay.

4. Chardonnay now receives an MPLS packet with a top label of 100,144, which it originally allocated from its label space for routes belonging to the VRF table. The table lookup tells Chardonnay to pop the top label and forward the remaining data, an IP packet in this case, out its `so-0/1/0.0` interface to Shiraz:

```
user@Chardonnay> show route table mpls.0 label 100144

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

100144             *[VPN/170] 1d 21:00:48
                    > to 10.222.44.1 via so-0/1/0.0, Pop
```

5. Shiraz, the remote CE router, receives the IP packet and finds that the 172.16.2.1 address is a local address, which makes it the destination of the packet. An ICMP echo reply message is formed by Shiraz and forwarded back to Chardonnay, where the same MPLS forwarding process occurs in reverse:

```
user@Shiraz> show route 172.16.2.1

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

172.16.2.1/32      *[Direct/0] 1d 21:03:35
                    > via lo0.0
```

To see the end result of the forwarding process, we use the `ping` command on Sangiovese to generate some traffic within the VPN:

```
user@Sangiovese> ping 172.16.2.1 source 172.16.1.1
PING 172.16.2.1 (172.16.2.1): 56 data bytes
64 bytes from 172.16.2.1: icmp_seq=0 ttl=253 time=1.522 ms
64 bytes from 172.16.2.1: icmp_seq=1 ttl=253 time=1.254 ms
64 bytes from 172.16.2.1: icmp_seq=2 ttl=253 time=1.261 ms
^C
--- 172.16.2.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.254/1.346/1.522/0.125 ms
```

NOTE

The use of the `source` option alters the source IP address of the ICMP packets. This is critical in our example because Shiraz has a route for the 172.16.1.0 /24 subnet only and not the 10.222.30.0 /24 subnet.

Using BGP for PE-CE Route Advertisements

While our sample Layer 3 VPN in Figure 9.1 works quite well using static routes, many network administrators prefer the dynamic nature of a routing protocol. The JUNOS software allows us to use RIP, OSPF, and BGP as routing protocols for the CE-PE connection. Within this section, we focus our attention on BGP. In our sample network, we migrate the current configurations from static routes to BGP. The CE router of Sangiovese is configured within Autonomous System 65000 and forms an EBGP peering session with its connected PE router of Chianti:

```
user@Sangiovese> show configuration routing-options
static {
    route 172.16.1.0/24 reject;
}
autonomous-system 65000;

user@Sangiovese> show configuration protocols bgp
group VPN-Connectivity {
    type external;
    export send-statics;
    peer-as 65432;
    neighbor 10.222.30.2;
}

user@Sangiovese> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0                 1          1          0          0          0          0
Peer               AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State
10.222.30.2     65432         15         17       0       0        5:38 1/1/0
```

The other CE router, Shiraz, is configured in a similar manner. Each of the CE routers advertises a static route to its EBGP peer. This route represents connectivity throughout its site to its remote CE partner:

```
user@Sangiovese> show route advertising-protocol bgp 10.222.30.2 172.16/16

inet.0: 14 destinations, 14 routes (14 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.0/24           Self                 0                  I

user@Shiraz> show route advertising-protocol bgp 10.222.44.2 172.16/16
```

```
inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.2.0/24           Self                 0                  I
```

To ensure that the VPN is working correctly, let's focus on just the 172.16.1.0 /24 route advertised from Sangiovese. This route is received by Chianti within the VRF table and is currently an active route:

```
user@Chianti> show route table VPN-A 172.16.1/24

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

172.16.1.0/24      *[BGP/170] 00:10:08, MED 0, localpref 100
                      AS path: 65000 I
                    > to 10.222.30.1 via so-0/1/2.0
```

Because Chianti is using the automatic export of all VRF routes via the `vrf-target` command, the route is allocated a VPN label and advertised to Chardonnay with a route target of `target:65432:1111`:

```
user@Chianti> show route advertising-protocol bgp 192.168.32.1 detail

VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 10.222.30.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 192.168.20.1:3
     VPN Label: 100384
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111

* 172.16.1.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 192.168.20.1:3
     VPN Label: 100416
     Nexthop: Self
     MED: 0
     Localpref: 100
     AS path: 65000 I
     Communities: target:65432:1111
```

The listing of `Nexthop: Self` within the router output tells us that the BGP Next Hop is set to 192.168.20.1, which is a valid address within the `inet.3` routing table on Chardonnay:

```
user@Chardonnay> show route table inet.3 192.168.20.1

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

192.168.20.1/32    *[RSVP/7] 3d 02:40:50, metric 20
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti
```

The Chardonnay router is also configured to accept all routes with a route target of `target:65432:1111`, which installs the 172.16.1.0 /24 route in both the `bgp.l3vpn.0` and the `VPN-A.inet.0` routing tables:

```
user@Chardonnay> show route table bgp.l3vpn.0

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.20.1:3:10.222.30.0/24
                   *[BGP/170] 00:15:27, localpref 100, from 192.168.20.1
                      AS path: I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti
192.168.20.1:3:172.16.1.0/24
                   *[BGP/170] 00:15:27, MED 0, localpref 100, from 192.168.20.1
                      AS path: 65000 I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti

user@Chardonnay> show route table VPN-A 172.16.1/24

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

172.16.1.0/24      *[BGP/170] 00:15:40, MED 0, localpref 100, from 192.168.20.1
                      AS path: 65000 I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti
```

The presence of a valid EBGP peering session between Chardonnay and Shiraz allows the advertisement of the 172.16.1.0 /24 route to Shiraz:

```
user@Chardonnay> show route advertising-protocol bgp 10.222.44.1
```

```
VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
* 10.222.30.0/24         Self                                    I
* 172.16.1.0/24          Self                                    65000 I
* 172.16.2.0/24          10.222.44.1                             65000 I
```

An examination of the routing table on the Shiraz router reveals a problem, however. The route is no longer visible:

```
user@Shiraz> show route 172.16.1/24

user@Shiraz> show route 172.16.1/24 all

user@Shiraz> show route receive-protocol bgp 10.222.44.2

inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
* 10.222.30.0/24         10.222.44.2                             65432 I
```

While we've experienced this problem before in our discussion of BGP, it's helpful to revisit the behavior we're seeing. It appears as if the Chardonnay router is advertising the route but the Shiraz router is not receiving it. A closer examination of the routes advertised by Chardonnay reveals an important clue:

```
user@Chardonnay> show route advertising-protocol bgp 10.222.44.1 detail

VPN-A.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 10.222.30.0/24 (1 entry, 1 announced)
 BGP group Shiraz-CE type External
     Nexthop: Self
     AS path: I
     Communities: target:65432:1111

* 172.16.1.0/24 (1 entry, 1 announced)
 BGP group Shiraz-CE type External
     Nexthop: Self
     AS path: 65000 I
     Communities: target:65432:1111

* 172.16.2.0/24 (1 entry, 1 announced)
 BGP group Shiraz-CE type External
     Nexthop: 10.222.44.1
     AS path: 65000 I
```

When Chardonnay advertises the route across its EBGP peering session with Shiraz, it prepends its local AS value of 65432 to the AS Path attribute. However, the attribute currently contains the AS value of 65000, which Sangiovese placed there during its advertisement to Chianti. When Shiraz receives the route with an AS Path of 65432 65000, it rejects the route as a routing loop prevention mechanism. The resolution most commonly applied in this situation is the configuration of the `as-override` command within the routing instance configuration of the PE routers. We update Chardonnay's configuration like this:

```
user@Chardonnay> show configuration routing-instances protocols bgp
group Shiraz-CE {
    type external;
    peer-as 65000;
    as-override;
    neighbor 10.222.44.1;
}
```

This configuration allows Chardonnay to examine the contents of the AS Path attribute to determine if its peer's AS value appears within the path. If it finds the peer's AS value, the local router replaces it with its own local AS value. This allows Shiraz to receive the 172.16.1.0 /24 route with an AS Path of 65432 65432 and accept the route:

```
user@Shiraz> show route 172.16.1/24

inet.0: 13 destinations, 14 routes (13 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.1.0/24      *[BGP/170] 00:03:27, localpref 100
                      AS path: 65432 65432 I
                    > to 10.222.44.2 via so-0/1/0.0
```

After applying the `as-override` command on Chianti, the Sangiovese router is able to receive and install the 172.16.2.0 /24 route:

```
user@Sangiovese> show route 172.16.2/24

inet.0: 15 destinations, 16 routes (15 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.2.0/24      *[BGP/170] 00:00:05, localpref 100
                      AS path: 65432 65432 I
                    > to 10.222.30.2 via so-0/2/0.0
```

With each of the CE routers installing a route to its partner, we can verify connectivity with a simple `ping` command:

```
user@Sangiovese> ping 172.16.2.1
PING 172.16.2.1 (172.16.2.1): 56 data bytes
64 bytes from 172.16.2.1: icmp_seq=0 ttl=253 time=1.474 ms
64 bytes from 172.16.2.1: icmp_seq=1 ttl=253 time=1.234 ms
64 bytes from 172.16.2.1: icmp_seq=2 ttl=253 time=1.229 ms
^C
--- 172.16.2.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.229/1.312/1.474/0.114 ms
```

Using OSPF for PE-CE Route Advertisements

When a customer is replacing an existing set of wide area links for a Layer 3 VPN, it often wishes to continue using the IGP in place before the switchover. One very popular customer routing protocol is OSPF, so let's see how this operates within our environment.

Within Figure 9.1, we now examine the operation of a VPN on the other half of our provider network. The Sherry and Chablis routers are CE devices using OSPF as their IGP for inter-site connectivity. Zinfandel is acting as a P router to connect the PE routers of Chianti and Cabernet. Both of the PE routers are using the automatic policy available through the `vrf-target` command. As an example, the Chianti router's configuration is

```
user@Chianti> show configuration routing-instances VPN-B
instance-type vrf;
interface ge-0/2/0.0;
route-distinguisher 65432:2222;
vrf-target target:65432:2222;
protocols {
    ospf {
        area 0.0.0.0 {
            interface ge-0/2/0.0;
        }
    }
}
```

The route target for the VRF is set to `target:65432:2222`. This AS-based value is also manually set as the RD within the routing instance. While this commonality is not a requirement, it makes operating and troubleshooting the VRF table easier.

At this point, each of the CE routers is maintaining a `Full` OSPF adjacency and is advertising their loopback address to their peer. To illustrate the VPN operation, let's focus on the 192.168.16.1 address of the Sherry router. This appears within the VRF table on Chianti as an OSPF route:

```
user@Chianti> show route protocol ospf table VPN-B.inet.0

VPN-B.inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.16.1/32    *[OSPF/10] 00:21:35, metric 1
                    > to 10.222.29.1 via ge-0/2/0.0
224.0.0.5/32       *[OSPF/10] 00:21:50, metric 1
                      MultiRecv
```

The automatic VRF export policy configured on Chianti allows for the allocation of a VPN label. This label, as well as the route target, is advertised to the remote PE router of Cabernet:

```
user@Chianti> ...rotocol bgp 192.168.48.1 192.168/16 detail

VPN-B.inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
* 192.168.16.1/32 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 65432:2222
     VPN Label: 100448
     Nexthop: Self
     MED: 1
     Localpref: 100
     AS path: I
     Communities: target:65432:2222 rte-type:0.0.0.0:1:0
```

The BGP Next Hop is currently set to 192.168.20.1. This address is visible within the `inet.3` routing table on the Cabernet router:

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

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

192.168.20.1/32    *[RSVP/7] 01:04:03, metric 20
                    > via so-0/1/0.0, label-switched-path Cab-to-Chianti
192.168.32.1/32    *[RSVP/7] 3d 04:11:40, metric 10
                    > via so-0/1/1.0, label-switched-path Cab-to-Char
```

The automatic VRF import policy on Cabernet installs the route in the VPN-A.inet.0 routing table:

```
user@Cabernet> show route 192.168.16/24 table VPN-B.inet.0

VPN-B.inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.16.1/32    *[BGP/170] 00:27:09, MED 1, localpref 100, from 192.168.20.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Cab-to-Chianti
```

When we look at the routing table on the Chablis router, however, we find that the route doesn't exist:

```
user@Chablis> show route 192.168.16/24

user@Chablis> show route 192.168.16/24 all

user@Chablis>
```

In fact, this situation is quite normal when combining routing knowledge from multiple protocols. The only method for advertising routes from Cabernet to Chablis is via OSPF, but the route is currently installed within the VRF table on Cabernet as a BGP route. This sounds like a job for route redistribution and a routing policy, so let's create the ***bgp-to-ospf*** policy on Cabernet and apply it within the routing instance:

```
user@Cabernet> show configuration policy-options
policy-statement bgp-to-ospf {
    term advertise-remote-routes {
        from protocol bgp;
        then accept;
    }
}

user@Cabernet> show configuration routing-instances VPN-B protocols
ospf {
    export bgp-to-ospf;
    area 0.0.0.0 {
        interface so-0/1/2.0;
    }
}
```

This allows Cabernet to advertise the 192.168.16.1 /32 route within an OSPF link-state advertisement to Chablis. The route then appears within the routing table on the CE router:

```
user@Chablis> show ospf database netsummary

    OSPF link state database, area 0.0.0.0
 Type       ID              Adv Rtr           Seq      Age  Opt  Cksum  Len
Summary  192.168.16.1     10.222.60.2      0x80000001    65  0x82 0x7ebb  28

user@Chablis> show route 192.168.16.1

inet.0: 16 destinations, 17 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.16.1/32    *[OSPF/10] 00:01:27, metric 2
                    > via so-0/1/0.0
```

Once we configure a similar policy on the Chianti router, the loopback address of Chablis (192.168.52.1) appears in the routing table on Sherry:

```
user@Sherry> show route 192.168.52.1

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

192.168.52.1/32    *[OSPF/10] 00:00:04, metric 2
                    > to 10.222.29.2 via ge-0/2/0.0
```

We've now established connectivity between the loopback addresses of the two CE routers:

```
user@Sherry> ping 192.168.52.1 source 192.168.16.1
PING 192.168.52.1 (192.168.52.1): 56 data bytes
64 bytes from 192.168.52.1: icmp_seq=0 ttl=253 time=1.129 ms
64 bytes from 192.168.52.1: icmp_seq=1 ttl=253 time=0.903 ms
64 bytes from 192.168.52.1: icmp_seq=2 ttl=253 time=0.897 ms
^C
--- 192.168.52.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.897/0.976/1.129/0.108 ms
```

Domain ID

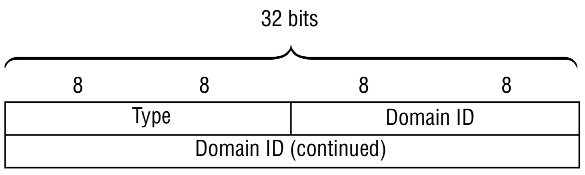
You might have noticed in the previous section that Sherry received the 192.168.52.1 /32 route in a Summary Type 3 LSA from Chianti. In a normal route redistribution environment, the ***bgp-to-ospf*** policy on Chianti injects routes into OSPF as AS External Type 5 LSAs. Within a VPN environment, this behavior of injecting external routes may not be desirable. For example, suppose that a customer is migrating from an existing WAN infrastructure to a Layer 3 VPN. The customer wants to bring up the VPN links between sites individually and begin using them for data transport before removing the existing connectivity. In this environment, the two sites are exchanging routes across the existing infrastructure using Summary LSAs. The same routes are also advertised across the VPN link but are advertised as AS External LSAs. Since internal OSPF routes are always more preferred to external routes, the remotes sites continue to communicate across the existing links.

To alleviate this dilemma, the JUNOS software assigns a *domain ID* value to each routing instance running OSPF. By default, this 32-bit value is set to all zeros (0.0.0.0). The domain ID allows the receiving PE router to advertise routes as either Type 3 or Type 5 LSAs. The format of the domain ID is shown in Figure 9.4.

Type (2 octets) This field displays the type of extended community advertised with the route. The valid type codes for the OSPF domain ID are 0x0005, 0x0105, and 0x0205. The high-order byte in this field technically determines the format of the remaining octets, but is irrelevant in our case since all the remaining octets compose the domain ID value.

Domain ID (6 octets) This field contains the domain ID value advertised by the PE router.

FIGURE 9.4 The OSPF domain ID



Before we examine the use of the domain ID in further detail, we also need to examine a BGP extended community called the *OSPF route type attribute*. This attribute is included automatically by the JUNOS software in all advertised OSPF routes and interacts with the domain ID on the receiving PE router to determine the type of route advertised. Figure 9.5 displays the format of the OSPF route type attribute, which includes the following fields:

Type (2 octets) This field displays the type of extended community advertised with the route. The OSPF route type attribute uses a constant value of 0x0306 as its type code.

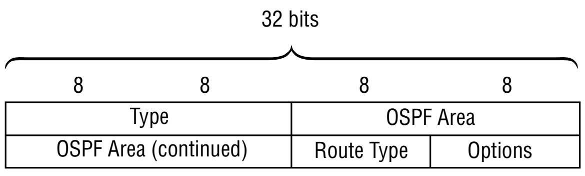
OSPF Area (4 octets) This field contains the OSPF area value operating on the advertising PE router that received the original OSPF route. For received external routes, this field is set to all zeros (0.0.0.0).

Route Type (1 octet) The route type field displays the LSA type of the route being advertised. It represents the method by which the advertising PE router learned the OSPF route. The possible values include

- 1—Intra-area routes from a Router LSA
- 2—Intra-area routes from a Network LSA
- 3—Inter-area routes from a Summary LSA
- 5—External routes from an AS External LSA
- 7—External routes from an NSSA LSA

Options (1 octet) This field is used only when the route type is set to a value of 5 or 7. The least significant bit in the field is set to a 1 (0x01) when the external route has a type 2 metric. This bit is set to a 0 (0x00) when the external route has a type 1 metric.

FIGURE 9.5 The OSPF route type attribute



An OSPF route advertised across a VPN connection always contains the route type attribute but may not contain a domain ID. The absence of an ID value is interpreted by a receiving PE router as a null domain ID (0.0.0.0), the default JUNOS software value. We see these conditions on the Chianti router in Figure 9.1 as it receives the 192.168.52.1 /32 route from Cabernet:

```
user@Chianti> ...ute table bgp.l3vpn.0 source-gateway 192.168.48.1 detail

bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
65432:2222:10.222.60.0/24 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65432:2222
                Source: 192.168.48.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Cab
                Label operation: Push 100368, Push 100464(top)
                Protocol next hop: 192.168.48.1
                Push 100368
                 Indirect next hop: 85222a0 336
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 1:20       Metric2: 20
                Task: BGP_65432.192.168.48.1+4403
                AS path: I
                Communities: target:65432:2222
                VPN Label: 100368
                Localpref: 100
                Router ID: 192.168.48.1
                Secondary Tables: VPN-B.inet.0
```

```
65432:2222:192.168.52.1/32 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65432:2222
                Source: 192.168.48.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Cab
                Label operation: Push 100416, Push 100464(top)
                Protocol next hop: 192.168.48.1
                Push 100416
                 Indirect next hop: 85223f0 344
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 1:20       Metric: 1       Metric2: 20
                Task: BGP_65432.192.168.48.1+4403
                AS path: I
                Communities: target:65432:2222 rte-type:0.0.0.0:1:0
                VPN Label: 100416
                Localpref: 100
                Router ID: 192.168.48.1
                Secondary Tables: VPN-B.inet.0
```

The attribute value tells us that Cabernet received the route on an area 0 interface as a Router LSA (Type 1). The absence of a domain ID community means that Cabernet is using the default null ID value of all zeros. As the route contains the proper route target for the VPN, Chianti needs to advertise the route to its CE router of Sherry. It uses the route type and domain ID values in the following manner to determine if the route should be advertised as a Summary or an AS External LSA:

1. When the received route has an internal route type (1, 2, or 3) but does not contain a domain ID attribute, it is advertised as a Type 3 Summary LSA.
2. When the received route has an internal route type and a domain ID value equal to the locally configured domain ID, it is advertised as a Type 3 Summary LSA.
3. When the received route has an internal route type and a domain ID that doesn't equal the locally configured domain ID, it is advertised as a Type 5 AS External LSA.
4. When the received route has an internal route type and a domain ID value attached, but the local router has no configured ID value (the default null ID), it is advertised as a Type 5 AS External LSA.
5. When the received route has an external route type (5 or 7), it is always advertised as a Type 5 AS External LSA. This occurs regardless of the domain ID values contained on the local router and the received route.

Let's see how the JUNOS software interacts with these advertisement rules using the routers in Figure 9.1. The PE routers are each configured with a domain ID value of 1.1.1.1. Cabernet now looks like this:

```
user@Cabernet> show configuration routing-instances VPN-B protocols
ospf {
    domain-id 1.1.1.1;
    export bgp-to-ospf;
    area 0.0.0.0 {
        interface so-0/1/2.0;
    }
}
```

We create a routing policy called ***vpn-b-export***, which accepts all OSPF routes within the VRF table on Cabernet and applies the appropriate route target as well as the OSPF domain ID of 1.1.1.1. The policy also accepts all direct routes (the CE-PE link) and applies the route target while rejecting all other routes:

```
user@Cabernet> show configuration policy-options
policy-statement vpn-b-export {
    term send-local-ospf-routes {
        from protocol ospf;
        then {
            community add ospf-domain-id;
            community add vpn-b-route-target;
            accept;
        }
    }
    term send-direct-routes {
        from protocol direct;
        then {
            community add vpn-b-route-target;
            accept;
        }
    }
    term reject-all-else {
        then reject;
    }
}
community ospf-domain-id members domain:1.1.1.1:0;
community vpn-b-route-target members target:65432:2222;
```

The policy is applied to the routing instance on Cabernet using the `vrf-export` command. While this supersedes the settings within the `vrf-target` configuration for exported routes, we leave the `vrf-target` in the instance to accept all received routes with the appropriate route target:

```
user@Cabernet> show configuration routing-instances VPN-B
instance-type vrf;
interface so-0/1/2.0;
route-distinguisher 65432:2222;
vrf-export vpn-b-export;
vrf-target target:65432:2222;
protocols {
    ospf {
        domain-id 1.1.1.1;
        export bgp-to-ospf;
        area 0.0.0.0 {
            interface so-0/1/2.0;
        }
    }
}
```

The 192.168.52.1 /32 route now appears on the Chianti router with multiple BGP extended communities attached:

```
user@Chianti> ...ute table bgp.l3vpn.0 source-gateway 192.168.48.1 detail

bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
65432:2222:10.222.60.0/24 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65432:2222
                Source: 192.168.48.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Cab
                Label operation: Push 100368, Push 100464(top)
                Protocol next hop: 192.168.48.1
                Push 100368
                 Indirect next hop: 85222a0 336
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 9:30       Metric2: 20
                Task: BGP_65432.192.168.48.1+4403
                AS path: I
                Communities: target:65432:2222
                VPN Label: 100368
                Localpref: 100
                Router ID: 192.168.48.1
                Secondary Tables: VPN-B.inet.0

65432:2222:192.168.52.1/32 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65432:2222
                Source: 192.168.48.1
                Next hop: via so-0/1/0.0 weight 1, selected
                Label-switched-path Chianti-to-Cab
                Label operation: Push 100432, Push 100464(top)
                Protocol next hop: 192.168.48.1
                Push 100432
                 Indirect next hop: 85223f0 344
                State: <Active Int Ext>
                Local AS: 65432 Peer AS: 65432
                Age: 8:02       Metric: 1       Metric2: 20
                Task: BGP_65432.192.168.48.1+4403
                AS path: I
                Communities: target:65432:2222 domain-id:1.1.1.1:0
                             rte-type:0.0.0.0:1:0
                VPN Label: 100432
                Localpref: 100
                Router ID: 192.168.48.1
                Secondary Tables: VPN-B.inet.0
```

The 192.168.52.1 /32 route has an OSPF route type of 1 and a matching domain ID to the local router. This allows Chianti to advertise the route as a Summary LSA, which we see in the link-state database on the Sherry router:

```
user@Sherry> show ospf database lsa-id 192.168.52.1

    OSPF link state database, area 0.0.0.0
 Type       ID               Adv Rtr          Seq      Age  Opt  Cksum  Len
Summary  192.168.52.1     10.222.29.2      0x80000002   373  0x82 0xc86b  28
```

We now change the domain ID configured on the Chianti router to a value of 2.2.2.2:

```
user@Chianti> show configuration routing-instances VPN-B protocols
ospf {
    domain-id 2.2.2.2;
    export bgp-to-ospf;
    area 0.0.0.0 {
        interface ge-0/2/0.0;
    }
}
```

The local domain ID on the receiving PE router no longer matches the value attached to the route—`domain-id:1.1.1.1:0`. Although the OSPF route type of the route is set to an internal value of 1, the domain ID mismatch causes Chianti to advertise the route as a Type 5 LSA. We again verify this advertisement by using the database on the Sherry router:

```
user@Sherry> show ospf database lsa-id 192.168.52.1

    OSPF AS SCOPE link state database
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern   192.168.52.1     10.222.29.2      0x80000001   137  0x2  0x3d05  36
```

In fact, we see this same Type 5 LSA when Chianti has no domain ID value configured. In essence, the default null value of 0.0.0.0 doesn't match the received domain ID of 1.1.1.1 attached to the route:

```
user@Chianti> show configuration routing-instances VPN-B protocols
ospf {
    export bgp-to-ospf;
    area 0.0.0.0 {
        interface ge-0/2/0.0;
    }
}

user@Sherry> show ospf database lsa-id 192.168.52.1

    OSPF AS SCOPE link state database
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern   192.168.52.1     10.222.29.2      0x80000001   264  0x2  0x3d05  36
```

The CE router of Chablis now advertises the 10.222.55.0 /24 route to Cabernet as an external OSPF route. We see this appear in the link-state database associated with the VRF table on Cabernet:

```
user@Cabernet> show ospf database instance VPN-B extern

    OSPF AS SCOPE link state database
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern  *10.222.29.0      10.222.60.2      0x8000009d   693  0x2  0x6837  36
Extern   10.222.55.0      192.168.52.1     0x80000001    36  0x2  0xd728  36
```

This route is advertised with an OSPF route type of 5, an external route, to Chianti:

```
user@Cabernet> show route advertising-protocol bgp 192.168.20.1 detail

VPN-B.inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
* 10.222.55.0/24 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 65432:2222
     VPN Label: 100432
     Nexthop: Self
     MED: 20
     Localpref: 100
     AS path: I
     Communities: target:65432:2222 domain-id:1.1.1.1:0 rte-type:0.0.0.0:5:1

* 10.222.60.0/24 (2 entries, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 65432:2222
     VPN Label: 100368
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:2222

* 192.168.52.1/32 (1 entry, 1 announced)
 BGP group int type Internal
     Route Distinguisher: 65432:2222
     VPN Label: 100432
     Nexthop: Self
     MED: 1
     Localpref: 100
     AS path: I
     Communities: target:65432:2222 domain-id:1.1.1.1:0 rte-type:0.0.0.0:1:0
```

The presence of this route type automatically prompts Chianti to advertise the route using an AS External Type 5 LSA, which is placed in the link-state database on Sherry:

```
user@Sherry> show ospf database extern

    OSPF AS SCOPE link state database
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern   10.222.55.0      10.222.29.2      0x80000001   260  0x2  0x2508  36
Extern   10.222.60.0      10.222.29.2      0x80000075   205  0x2  0x3c8b  36
Extern   192.168.52.1     10.222.29.2      0x80000002   505  0x2  0x3b06  36
```

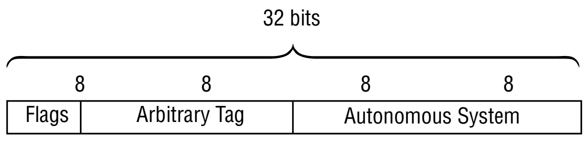
The VPN Route Tag

Internal OSPF routes have inherent routing loop prevention mechanisms in place on the area border routers, which we discussed in Chapter 2, "Open Shortest Path First." Unfortunately, no such process exists for Type 5 external routes because they are flooded to all possible routers by default. This domain-wide flooding process presents a problem in a Layer 3 VPN environment. Suppose that both Router A and Router B are PE routers that connect to different CE routers in a single customer site. When Router A receives an external route from a remote PE neighbor, which is not Router B, it redistributes that route to the customer site as a Type 5 LSA. Its neighboring CE router receives the LSA and floods it throughout the customer site, where it arrives at the CE router connected to Router B. By default, Router B receives the LSA within its VRF table and forwards it across the VPN to neighboring PE routers. This advertisement by Router B can potentially cause a routing loop in the network. To avoid such occurrences, each PE router using OSPF as a CE-PE routing protocol generates a unique *VPN route tag* value and places it within the Type 5 LSA. When a PE router receives an external advertisement from a CE router with a tag value matching the local tag value, the router doesn't advertise that route across the VPN. This prevents potential routing loops from forming in the network.

The JUNOS software automatically computes a VPN route tag for each routing instance running OSPF. The tag value is based on the Autonomous System value configured on the PE router using the format shown in Figure 9.6. The tag's fields include:

Flags (4 bits) The Flags field contains three attributes and is set to a value of 1101 by default. The most significant bit is set to a value of 1 to indicate that the route tag has been automatically calculated. The next bit in the field is also set to a value of 1 and informs all PE routers that this is the complete route tag information. The last 2 bits in the field are used to indicate the number of the AS values contained within the tag. These bits are set to a value of 01.

FIGURE 9.6 The OSPF VPN route tag



Arbitrary Tag (12 bits) This field is set to all zeros (0x000) when the PE router automatically calculates the VPN route tag.

Autonomous System (2 octets) The PE router places its own AS number in this field.

The Chianti router, which is within AS 65432, automatically generates a VPN route tag of `0xd000ff98` for the routing instance belonging to the Sherry router. When we change this hexadecimal value to a dotted decimal format, we find that the VPN route tag for the instance is 208.0.255.152. This value is placed within the Type 5 LSAs generated by Chianti:

```
user@Chianti> show ospf database instance VPN-B extern extensive
    OSPF AS SCOPE link state database
```

```
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern  *10.222.55.0      10.222.29.2      0x80000003    71  0x2  0x210a  36
  mask 255.255.255.0
  Type 2, TOS 0x0, metric 20, fwd addr 0.0.0.0, tag 208.0.255.152
  Gen timer 00:48:48
  Aging timer 00:58:48
  Installed 00:01:11 ago, expires in 00:58:49, sent 00:01:09 ago
  Ours
Extern  *10.222.60.0      10.222.29.2      0x80000076   971  0x2  0x3a8c  36
  mask 255.255.255.0
  Type 2, TOS 0x0, metric 0, fwd addr 0.0.0.0, tag 208.0.255.152
  Gen timer 00:10:03
  Aging timer 00:43:48
  Installed 00:16:11 ago, expires in 00:43:49, sent 00:16:09 ago
  Ours
Extern  *192.168.52.1     10.222.29.2      0x80000004   371  0x2  0x3708  36
  mask 255.255.255.255
  Type 1, TOS 0x0, metric 1, fwd addr 0.0.0.0, tag 208.0.255.152
  Gen timer 00:43:48
  Aging timer 00:53:48
  Installed 00:06:11 ago, expires in 00:53:49, sent 00:06:09 ago
  Ours
```

The JUNOS software allows you to set a separate VPN route tag for each customer through the configuration of the `domain-vpn-tag`. This is configured within the routing instance for the customer and contains a 32-bit value used as the route tag. The Chianti router is configured with a tag value of 2,071,690,107. This translates to a dotted decimal format of 123.123.123.123, as seen in the OSPF database on the PE router:

```
user@Chianti> show configuration routing-instances VPN-B protocols
ospf {
    domain-vpn-tag 2071690107;
    export bgp-to-ospf;
    area 0.0.0.0 {
        interface ge-0/2/0.0;
    }
}

user@Chianti> show ospf database instance VPN-B extern extensive
    OSPF AS SCOPE link state database
 Type       ID               Adv Rtr           Seq      Age  Opt  Cksum  Len
Extern  *10.222.55.0      10.222.29.2      0x80000004    24  0x2  0xfda7  36
  mask 255.255.255.0
  Type 2, TOS 0x0, metric 20, fwd addr 0.0.0.0, tag 123.123.123.123
  Gen timer 00:49:35
  Aging timer 00:59:35
  Installed 00:00:24 ago, expires in 00:59:36, sent 00:00:22 ago
  Ours
Extern  *10.222.60.0      10.222.29.2      0x80000078    24  0x2  0x152b  36
  mask 255.255.255.0
  Type 2, TOS 0x0, metric 0, fwd addr 0.0.0.0, tag 123.123.123.123
  Gen timer 00:49:35
  Aging timer 00:59:35
  Installed 00:00:24 ago, expires in 00:59:36, sent 00:00:22 ago
  Ours
Extern  *192.168.52.1     10.222.29.2      0x80000005    24  0x2  0x14a5  36
  mask 255.255.255.255
  Type 1, TOS 0x0, metric 1, fwd addr 0.0.0.0, tag 123.123.123.123
  Gen timer 00:49:35
  Aging timer 00:59:35
  Installed 00:00:24 ago, expires in 00:59:36, sent 00:00:22 ago
  Ours
```

NOTE

The JUNOS software also uses the VPN route tag to prevent routing loops. When a PE router receives a Type 5 LSA from a CE router, it examines both the LSA ID and the tag value. If the LSA ID is not the local router but the tag values match, the local PE router assumes that a possible loop is forming. To avoid this, the PE doesn't include that LSA in its SPF calculation.

Internet Access for VPN Customers

When a provider's customer is using a Layer 3 VPN for intra-site connectivity, the customer often wants to use the provider's network for Internet access. There are three main categories for providing this type of service: access outside the confines of the VRF table, distributed access to each VPN site, and centralized access from a single VPN site. Generally speaking, each of the main access categories also has multiple variations included within it.

Independent Internet Access

The main point to remember concerning *independent Internet access* is that the routing tables on the PE router are never consulted for packet forwarding. This means that the CE router either forwards traffic directly to the Internet or uses the PE router as a Layer 2 forwarding device to an Internet-aware router.

Figure 9.7 shows the first of these examples, with the CE routers of Sangiovese and Shiraz each maintaining its own connections to the Internet. In this environment, Sangiovese sends all VPN packets to Chianti for forwarding to the remote CE router of Shiraz. All packets whose destination address does not fall within the VPN are forwarded by Sangiovese directly to the Internet. Of course, this type of connectivity doesn't provide any revenue to the provider. If a customer wishes to control its access in this manner, the provider can offer a Layer 2 logical circuit to the customer, as we see in Figure 9.8.

Within this independent Internet access example, the network provider is providing two logical circuits to the customer. One of the circuits is configured within the VRF table and is used by the customer for forwarding traffic within the VPN. The second logical circuit passes through the PE router at Layer 2 and terminates at another router in the provider's network—Merlot in our case. The customer sends Internet-bound traffic on this second logical circuit, and Merlot forwards that traffic based on its routing knowledge of Internet destinations.

FIGURE 9.7 Independent Internet access (1 of 2)

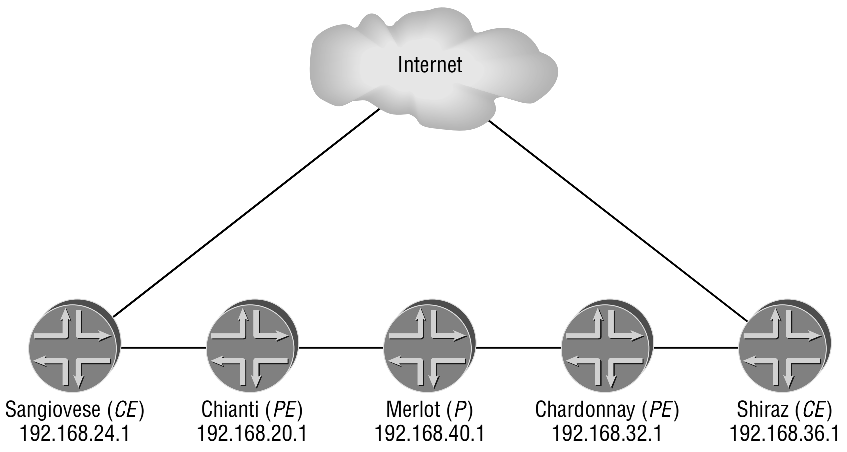
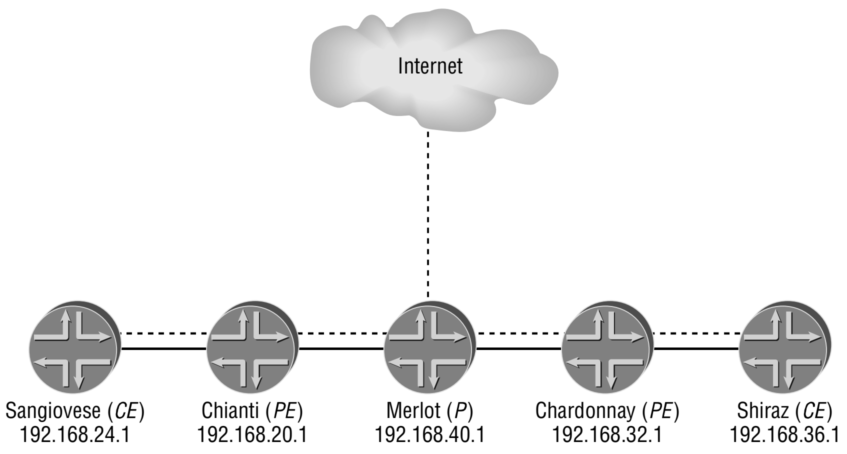

Internet
Sangiovese (CE) 192.168.24.1
Chianti (PE) 192.168.20.1
Merlot (P) 192.168.40.1
Chardonnay (PE) 192.168.32.1
Shiraz (CE) 192.168.36.1

FIGURE 9.8 Independent Internet access (2 of 2)

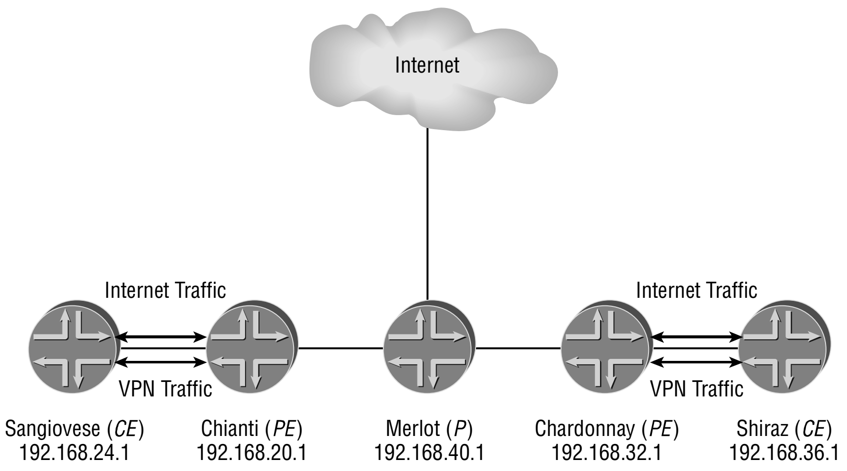

Internet
Sangiovese (CE) 192.168.24.1
Chianti (PE) 192.168.20.1
Merlot (P) 192.168.40.1
Chardonnay (PE) 192.168.32.1
Shiraz (CE) 192.168.36.1

Distributed Internet Access

In a *distributed Internet access* model, each and every PE router provides connectivity to the Internet for its connected CE router. Within this environment, the provider has three main options for forwarding traffic through the PE router.

The first option (Figure 9.9) involves two logical circuits between the PE and CE routers. One of the circuits is configured within the VRF table on Chianti and is used for connectivity within the VPN. Sangiovese uses this circuit for forwarding traffic to Shiraz. The second circuit is configured as a normal interface, and Sangiovese uses this connection for forwarding packets to the Internet. While the customer may choose to operate a routing protocol across the non-VRF interface, many providers use a static routing model in this situation.

FIGURE 9.9 Distributed Internet access (1 of 3)



The Sangiovese router has the following operational interfaces:

```
user@Sangiovese> show interfaces terse so-0/2/0
Interface               Admin Link Proto Local                 Remote
so-0/2/0                up    up
so-0/2/0.0              up    up   inet  10.222.30.1/24
so-0/2/0.1              up    up   inet  10.222.100.1/24
```

The `so-0/2/0.0` interface is communicating via a BGP peering with Chianti as part of the VRF table. This allows the VPN routes to appear in the routing table with a physical next-hop of 10.222.30.2:

```
user@Sangiovese> show route terse protocol bgp

inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 10.222.44.0/24     B 170        100            >10.222.30.2     65432 I
* 172.16.2.0/24      B 170        100            >10.222.30.2     65432 65432 I
```

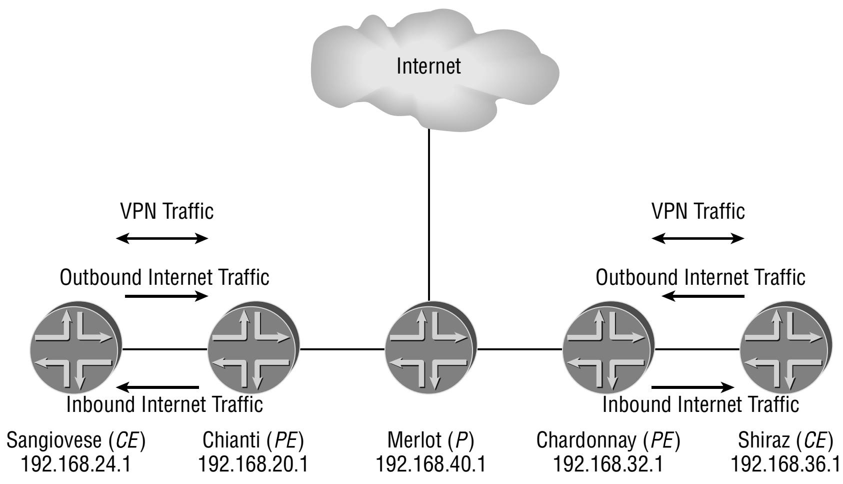
All Internet-bound traffic uses a next hop of 10.222.100.2 over the `so-0/1/0.1` interface:

```
user@Sangiovese> show route 0/0 exact

inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:02:29
                    > to 10.222.100.2 via so-0/2/0.1
```

While this route forwards traffic to Chianti and the Internet at large, we also need to be concerned about return traffic from the Internet. This reachability is easily solved by a static route on Chianti, which represents the customer's address space of 172.16.0.0/16. This route is redistributed to BGP and sent to the provider's network, where it is eventually advertised to the Internet:

```
user@Chianti> show route 172.16/16 exact

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

172.16.0.0/16      *[Static/5] 00:01:15
                    > to 10.222.100.1 via so-0/1/2.1
```

FIGURE 9.10 Distributed Internet access (2 of 3)



The second option for providing distributed Internet access to a customer (Figure 9.10) also involves two logical circuits between the PE and CE routers. The CE router sends both VPN and Internet traffic to the PE router across the same logical circuit. Return traffic from the Internet uses the second non-VRF logical circuit. From the perspective of Sangiovese, the CE router, both VPN and Internet routes have a next hop of 10.222.30.2 across the `so-0/1/2.0` interface:

```
user@Sangiovese> show route terse

inet.0: 18 destinations, 19 routes (18 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 0.0.0.0/0          S   5                       >10.222.30.2
* 1.1.1.0/24         D   0                       >fxp0.0
* 1.1.1.2/32         L   0                        Local
* 10.222.4.0/24      D   0                       >fe-0/0/2.0
* 10.222.4.1/32      L   0                        Local
* 10.222.28.0/24     D   0                       >at-0/1/0.0
* 10.222.28.2/32     L   0                        Local
* 10.222.30.0/24     D   0                       >so-0/2/0.0
* 10.222.30.1/32     L   0                        Local
* 10.222.44.0/24     B 170        100            >10.222.30.2     65432 I
* 10.222.100.0/24    D   0                       >so-0/2/0.1
* 10.222.100.1/32    L   0                        Local
* 10.250.0.0/16      D   0                       >fxp0.0
* 10.250.0.132/32    L   0                        Local
* 172.16.1.0/24      S   5          0             Reject
* 172.16.1.1/32      D   0                       >lo0.0
* 172.16.2.0/24      B 170        100            >10.222.30.2     65432 65432 I
* 192.168.24.1/32    D   0                       >lo0.0
```

The main difference in this scenario is apparent on the PE router of Chianti. Since Internet-bound packets from the customer are arriving in the VRF table, the PE router needs a route in that table that allows the destination to be found. All Internet destinations can be placed within the VRF table itself, but this presents a large scalability problem when many customers are connected to the same PE router. In addition, those destinations are most likely already present in the `inet.0` routing table on the PE router. This provides an interesting solution to our dilemma; now we need only a default route in the VRF table that allows for a second lookup in the main routing table. The JUNOS software uses a static route next-hop value of `next-table` to permit this "double lookup" on the router. The configuration of the routing instance on Chianti now looks like this:

```
user@Chianti> show configuration routing-instances VPN-A routing-options
static {
    route 0.0.0.0/0 next-table inet.0;
}
```

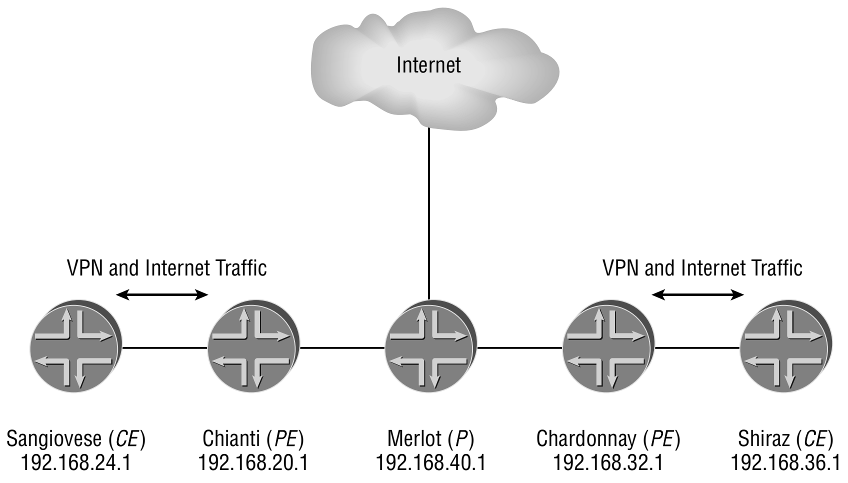
This allows the VRF table to contain a default route for matching destinations within the Internet:

```
user@Chianti> show route table VPN-A 0/0 exact

VPN-A.inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:01:16
                      to table inet.0
```

Of course, Chianti still requires a route for the customer's address space in `inet.0` for advertisement to BGP and to the Internet. This allows return traffic from the Internet to reach the customer routers:

```
user@Chianti> show route 172.16/16 exact

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

172.16.0.0/16      *[Static/5] 00:22:22
                    > to 10.222.100.1 via so-0/1/2.1
```

FIGURE 9.11 Distributed Internet access (3 of 3)



The final option for providing distributed Internet access for a customer uses just a single logical circuit between the CE and PE routers. In Figure 9.11, the CE router sends all traffic to the PE router, where route lookups are performed within the VRF table. This requires a static default route in the VRF table for Internet traffic that points to the `inet.0` routing table using the `next-table` command. On the PE router, no static routes are used for advertising the customer routes to the Internet or for routing traffic back to the CE router. Instead, all of the routes received by the PE router in the VRF table are copied to `inet.0` using a routing table group.

The configuration of Chianti is altered to support this Internet access scenario:

```
user@Chianti> show configuration routing-options
rib-groups {
    vpn-a-into-inet-0 {
        import-rib [ VPN-A.inet.0 inet.0 ];
    }
}
route-distinguisher-id 192.168.20.1;
autonomous-system 65432;

user@Chianti> show configuration routing-instances VPN-A
instance-type vrf;
interface so-0/1/2.0;
vrf-target target:65432:1111;
routing-options {
    static {
        route 0.0.0.0/0 next-table inet.0;
    }
}
protocols {
    bgp {
        group Sangiovese-CE {
            type external;
            family inet {
                unicast {
                    rib-group vpn-a-into-inet-0;
                }
            }
            peer-as 65000;
            as-override;
            neighbor 10.222.30.1;
        }
    }
}
```

This allows the 172.16.1.0 /24 route to appear in `inet.0` with a next-hop value of 10.222.30.1, the interface address of Sangiovese:

```
user@Chianti> show route terse 172.16.1.0/24 table inet.0

inet.0: 18 destinations, 18 routes (17 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 172.16.1.0/24      B 170        100          0 >10.222.30.1     65000 I
```

Of course, the VRF table still contains the static default route referencing the `inet.0` routing table:

```
user@Chianti> show route table VPN-A.inet.0

VPN-A.inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:19:35
                      to table inet.0
10.222.30.0/24     *[Direct/0] 00:47:56
                    > via so-0/1/2.0
10.222.30.2/32     *[Local/0] 00:47:56
                      Local via so-0/1/2.0
10.222.44.0/24     *[BGP/170] 00:04:02, localpref 100, from 192.168.40.1
                      AS path: I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
172.16.1.0/24      *[BGP/170] 00:03:50, MED 0, localpref 100
                      AS path: 65000 I
                    > to 10.222.30.1 via so-0/1/2.0
172.16.2.0/24      *[BGP/170] 00:04:02, MED 0, localpref 100, from 192.168.40.1
                      AS path: 65000 I
                    > via so-0/1/0.0, label-switched-path Chianti-to-Char
```

We can also see from Chianti's router output that packets destined for customer sites within the VRF table are forwarded across the provider network to the remote CE router of Shiraz. An example of this is the 172.16.2.0 /24 route, with a forwarding next hop of the ***`Chianti-to-Char`*** LSP.

Centralized Internet Access

An ISP also has the option of providing its customers with access using a *centralized Internet access* model. The various methods for providing this service are identical to the distributed Internet access models we discussed in the previous section. The main difference between the two models is that just one CE-PE router pair is configured for Internet access. This central CE router advertises a 0.0.0.0 /0 default route to the other CE routers within the VPN. This route advertisement attracts Internet-bound traffic to the central CE router, which then forwards it to the Internet.

Using Figure 9.9 as a guide, we designate the CE router of Sangiovese as the central site that has Internet access configured. This router has a static default route installed that uses the non-VRF interface of `so-0/2/0.1` for forwarding Internet traffic to the PE router:

```
user@Sangiovese> show route 0/0 exact

inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:10:12
                    > to 10.222.100.2 via so-0/2/0.1
```

In order to attract traffic from the other CE routers in the VPN, Sangiovese advertises the default route to the PE router as a VPN route. This is in addition to the locally reachable address space of 172.16.1.0 /24:

```
user@Sangiovese> show route advertising-protocol bgp 10.222.30.2

inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 2 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0              Self                                    I
* 10.222.44.0/24         10.222.30.2                             65432 I
* 172.16.1.0/24          Self                 0                  I
* 172.16.2.0/24          10.222.30.2                             65432 65432 I
```

Shiraz, the other CE device in the VPN, now has a default route for forwarding Internet-bound packets:

```
user@Shiraz> show route 0/0 exact

inet.0: 14 destinations, 15 routes (14 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:13:39, localpref 100
                      AS path: 65432 65432 I
                    > to 10.222.44.2 via so-0/1/0.0
```

When the packets reach the PE router connected to Shiraz, which is Chardonnay, they are forwarded to Chianti across the ***Char-to-Chianti*** LSP:

```
user@Chardonnay> show route 0/0 exact

VPN-A.inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:14:38, localpref 100, from 192.168.40.1
                      AS path: 65000 I
                    > via so-0/1/1.0, label-switched-path Char-to-Chianti
```

The packets are received by Sangiovese on its `so-0/2/0.0` interface and are passed back to Chianti on the non-VRF interface of `so-0/2/0.1`. Chianti then forwards them to the Internet using information in its `inet.0` routing table.

Transporting Layer 2 Frames across a Provider Network

The concept of a Layer 2 VPN has been around for quite some time. Traditional X.25, Frame Relay, and ATM networks are prime examples of this connectivity paradigm. Each technology transfers Layer 2 frames that are unique to an individual customer across a common provider infrastructure. Of course, many providers built separate infrastructures to transport these different Layer 2 technologies across their networks. What modern Layer 2 VPNs bring to the table is the ability to transport all Layer 2 technologies across a common IP network using MPLS labels.

Within the JUNOS software, two main varieties of these VPNs exist. To help differentiate between the different Layer 2 VPNs, we'll use the configuration syntax as our guide. The first type of VPN is based on a draft specification by Kireeti Kompella. It uses the Border Gateway Protocol (BGP) as the mechanism for PE routers to communicate with each other about their customer connections. We'll refer to a Kompella-based configuration as a Layer 2 VPN. The second main form of a VPN is based on a draft specification by Luca Martini and uses the Label Distribution Protocol (LDP) between PE routers. Every router establishes a unique connection for each customer using the VPN. The Martini-based VPN is known as a Layer 2 Circuit within the configuration.

Figure 9.12 displays a provider network that we'll refer to throughout the remainder of the chapter to illustrate the operation of Layer 2 VPNs as well as Layer 2 circuits. This particular network is using IS-IS as its internal routing protocol, and each of the P and PE routers are advertising information to the network. We see this with a quick examination of the IS-IS database on the Zinfandel router:

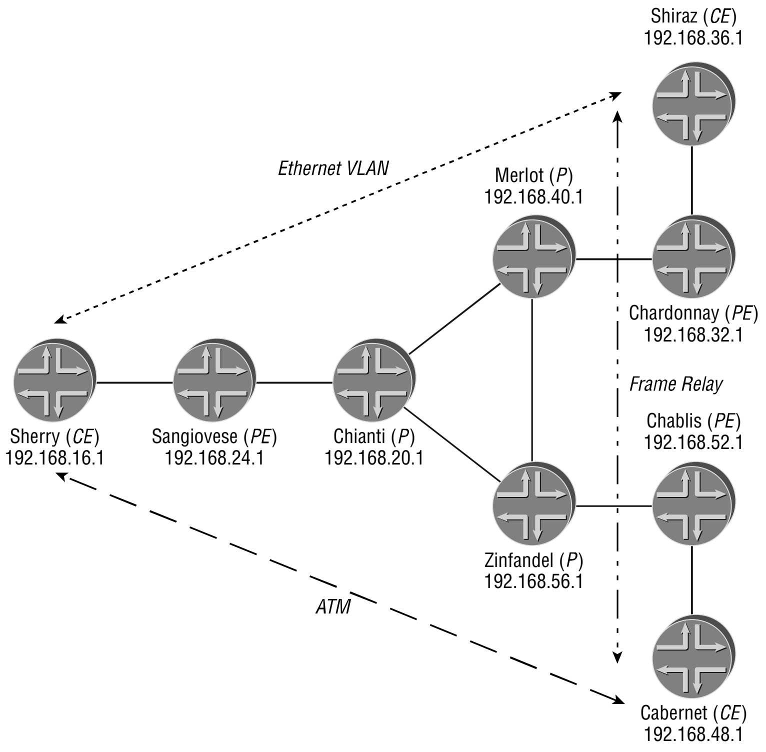
```
user@Zinfandel> show isis database level 2
IS-IS level 2 link-state database:
```

```
LSP ID                      Sequence Checksum Lifetime Attributes
Chianti.00-00                   0xdb   0xe634     1195 L1 L2
Sangiovese.00-00                0xdb   0x29f9      509 L1 L2
Chardonnay.00-00                0xd9   0x349a      627 L1 L2
Merlot.00-00                    0xdc   0xa077      442 L1 L2
Chablis.00-00                   0xdc   0xab3e      921 L1 L2
Zinfandel.00-00                 0xda   0xcff2      612 L1 L2
  6 LSPs
```

Layer 2 VPN

A Layer 2 VPN within the JUNOS software utilizes the same provider infrastructure as the Layer 3 VPNs we discussed earlier in the chapter. This means that the PE routers use MBGP to advertise information between them concerning their connected customers. These advertisements contain a VPN connection table (VCT) that describes label information used to connect the customer sites. In addition, the MBGP advertisements contain route target extended community values, which enforce the logical topology of the VPN.

FIGURE 9.12 Layer 2 VPN provider network



Each of the PE routers in Figure 9.12 has an active RSVP LSP from itself to other routers on the network. The Chardonnay router, for example, contains three LSPs:

```
user@Chardonnay> show mpls lsp ingress
Ingress LSP: 3 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.20.1    192.168.32.1    Up     0                  *     Char-to-Chianti
192.168.24.1    192.168.32.1    Up     0                  *     Char-to-Sang
192.168.52.1    192.168.32.1    Up     0                  *     Char-to-Chablis
Total 3 displayed, Up 3, Down 0
```

NOTE A Layer 2 VPN can also use LDP to provide MPLS connectivity between the PE routers.

To properly advertise the Layer 2 VPN information, each PE router is configured with the `family l2vpn unicast` command. This enables the negotiation of the Layer 2 VPN address family (196) as well as the labeled VPN unicast subsequent address family (128). The MBGP configuration of the Chardonnay router looks like this:

```
user@Chardonnay> show configuration protocols bgp
group internal-peers {
    type internal;
    local-address 192.168.32.1;
    family l2vpn {
        unicast;
    }
    neighbor 192.168.24.1;
    neighbor 192.168.20.1;
    neighbor 192.168.52.1;
}
```

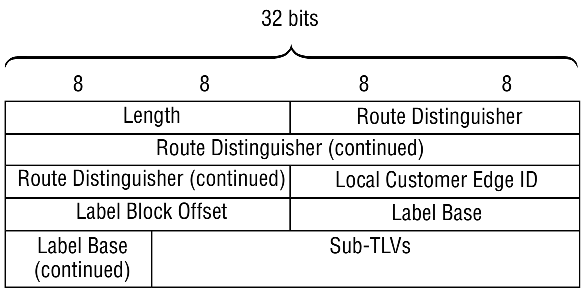
This allows the PE routers to advertise the Layer 2 VPN network layer reachability information (the VCT) to each of its peers. Figure 9.13 displays the format of this advertisement, which includes the following fields:

Length (2 octets) This field displays the total length, in octets, of the Layer 2 VPN information advertised by the PE router.

Route Distinguisher (8 octets) As we saw with a Layer 3 VPN, the Route Distinguisher field uniquely identifies each customer connection and allows the PE routers to maintain separation between the customer networks.

Local Customer Edge ID (2 octets) This field displays the *local customer edge ID* value assigned by the PE router to the customer connection. Each site within a particular customer VPN is assigned a unique ID value. This value is used to ensure that proper MPLS label values are used to send and receive traffic for a site.

FIGURE 9.13 Layer 2 VPN NLRI format



Label Block Offset (2 octets) A PE router has the ability to advertise multiple sets of labels to its remote PE peers. In this situation, each receiving PE router requires knowledge as to how the separate label blocks are sequenced. This field, the *label block offset*, provides this information. It contains a 16-bit value that allows the receiving routers to sequence the multiple advertised label blocks.

Label Base (3 octets) The Label Base field is used by PE routers to determine the exact MPLS label to use for forwarding and receiving traffic on a particular customer connection.

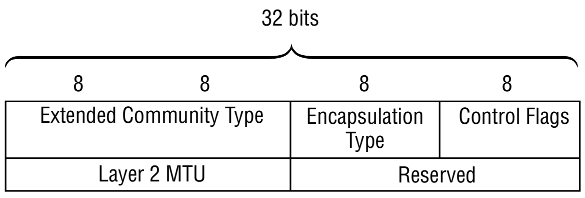
Sub-TLVs (Variable) This variable-length field contains sub-TLVs that further describe the customer connection. Each sub-TLV contains a 1-octet type field, a 2-octet length field, and a variable-length value field. Only a single sub-TLV, the circuit status vector, is currently defined.

The circuit status vector is a bit-vector system in which each of the configured customer connections is assigned a bit. In addition to allowing the PE routers to infer the range of labels used for the VPN, it is used to monitor the status of the virtual connection. Each PE router is aware of its local circuit and the LSP to the remote PE router. When both of these connections are operational, the local PE sets the appropriate bit in the vector to the value 0. When either of these connections stops working, the bit for that logical circuit is set to the value 1. This system allows the two PE routers to monitor the end-to-end customer circuit and notify the customer when the circuit is down. This notification comes in the form of a Layer 2 management mechanism, such as the Local Management Interface (LMI) for Frame Relay.

In addition to the route target community advertised along with the connection information, the Layer 2 VPN specifications define an additional community value. The *Layer 2 information community* provides a method to ensure that the two PE routers are connecting a logical circuit that is compatible between the customer connections. After all, if one customer is transmitting frames using a Frame Relay Data-Link Connection Identifier (DLCI), it doesn't do much good if the remote customer connection is expecting ATM cells. The fields of the Layer 2 information community are shown in Figure 9.14 and include:

Extended Community Type (2 octets) This field displays the type of extended community being advertised by the PE router. For the Layer 2 information community, a constant value of 0x800a is used.

FIGURE 9.14 Layer 2 information community



Encapsulation Type (1 octet) This field encodes the specific Layer 2 encapsulation used between the advertising PE router and its locally connected CE router. The possible values are as follows:

- 0—Reserved
- 1—Frame Relay
- 2—ATM AAL5 virtual circuit connection transport
- 3—ATM transparent cell transport
- 4—Ethernet virtual LAN
- 5—Ethernet
- 6—Cisco high-level data link control
- 7—Point-to-Point Protocol
- 8—Circuit emulation
- 9—ATM virtual circuit connection cell transport
- 10—ATM virtual path connection cell transport
- 11—IP Interworking
- 19—Virtual private label switching

Control Flags (1 octet) This field is a bit-vector that allows the advertising PE router to set parameters for the connection. The various bit definitions include:

Bits 7 through 4 These bits are not currently defined and must be set to a value of 0.

Bit 3 The Q bit is currently reserved and must be set to a value of 0.

Bit 2 The F bit is currently reserved and must be set to a value of 0.

Bit 1 The C bit defines the use of a *control word* within the provider network. The control word allows for the advertisement of Layer 2 management information between the customer sites by transporting it across the provider network. This is accomplished by adding a 4-byte header between the MPLS labels and the Layer 2 PDU at one edge of the provider network and further removing the control word at the other edge of the network.

When the C bit is set to a value of 1, the control word is required across the provider network. When it is set to a value of 0, it is not required.

Bit 0 The S bit defines whether a sequenced delivery of the Layer 2 frames is required. When the bit is set to a value of 1, the PE routers must ensure the sequencing of the transmitted frames. When it is set to a value of 0, sequenced delivery is not required.

Layer 2 MTU (2 octets) Each advertising PE router has the ability to set the current MTU, excluding the Layer 2 header information, of the customer connection in this 2-octet field. In addition, the field may be set to a value of 0 when the MTU information is not required. In either case, the MTU values must match between the two PE routers in order for the connection to become operational.

Reserved (2 octets) This field is not used and must be set to a constant value of 0x0000.

As you might expect, the configuration of a Layer 2 VPN is similar to a Layer 3 VPN. The main difference between the two is that the physical and logical interfaces connecting the PE router to the CE router requires some additional encapsulation information. We examine the specific details of the configuration within the context of a Frame Relay connection, but we cover both ATM and Ethernet configurations for completeness.

Frame Relay as the PE-CE Connection

One Layer 2 encapsulation that is currently popular as a wide area network technology is Frame Relay. This popularity makes it a leading candidate for migration to a Layer 2 VPN provider network. In addition to the establishment of the provider core network, which we discussed earlier in the chapter, the steps for enabling a Layer 2 VPN involve the encapsulation on the physical interfaces and the assignment of identifier values to the customer sites within the VPN.

Physical Interface Encapsulation

The end result of a Layer 2 VPN is the communication of two customer sites over a Frame Relay circuit. As such, the CE routers are configured to support Frame Relay and are assigned appropriate DLCI values by the network provider. Figure 9.12 shows a Frame Relay connection between Shiraz and Cabernet across our sample provider network. The interface configuration of these two routers currently looks like this:

```
user@Shiraz> show configuration interfaces so-0/1/0
encapsulation frame-relay;
unit 600 {
    dlci 600;
    family inet {
        address 10.222.100.1/24;
    }
}

user@Cabernet> show configuration interfaces so-0/1/2
encapsulation frame-relay;
unit 600 {
    dlci 600;
    family inet {
        address 10.222.100.2/24;
    }
}
```

The PE routers of Chardonnay and Chablis are responsible for receiving packets from and sending packets to the CE routers. As such, they also require some special interface configuration to support a Layer 2 VPN. Here's how they are configured:

```
user@Chardonnay> show configuration interfaces so-0/1/0
dce;
encapsulation frame-relay-ccc;
unit 600 {
    encapsulation frame-relay-ccc;
    dlci 600;
}

user@Chablis> show configuration interfaces so-0/1/0
dce;
encapsulation frame-relay-ccc;
unit 600 {
    encapsulation frame-relay-ccc;
    dlci 600;
}
```

NOTE For simplicity we've used the same DLCI value of 600 on both sides of the VPN, but this is not required. Only the PE and its connected CE need to agree on the DLCI used.

WARNING When you're using the `frame-relay-ccc` encapsulation, the DLCI values must be greater than or equal to 512.

The use of the `frame-relay-ccc` encapsulation on the PE routers allows the Layer 2 frames to be placed directly into an MPLS label-switched path for transmission across the network. It also provides the capability for the PE router to receive an MPLS packet and forward the encapsulated Layer 2 frame to the CE router.

After each PE-CE router pair begins exchanging the correct DLCI information, the interfaces transition to an `Up` state:

```
user@Chardonnay> show interfaces so-0/1/0 terse
Interface              Admin Link Proto Local                 Remote
```

```
so-0/1/0                up    up
so-0/1/0.600            up    up   ccc

user@Chablis> show interfaces so-0/1/0 terse
Interface               Admin Link Proto Local                 Remote
so-0/1/0                up    up
so-0/1/0.600            up    up   ccc
```

Configuring Site Identifiers

You may have noticed that the NLRI advertised in a Layer 2 VPN doesn't contain an explicit VPN label for use by the remote PE router. In its place are an advertised label base and a label offset value. This allows the receiving PE router to automatically calculate the VPN label used to reach the remote site based on its local configuration. This local configuration centers on the *site identifier* for the customer.

Each customer site connected to a PE router is assigned a site ID within the VPN using the `site-identifier` command. This 16-bit value is required to be unique only within the particular customer network. In our sample network the Shiraz router is assigned a site ID of 1, whereas the Cabernet router is assigned a site ID of 4. The configuration of the Chardonnay router, which is connected to Shiraz, looks like this:

```
user@Chardonnay> show configuration routing-instances
FR-Customer {
    instance-type l2vpn;
    interface so-0/1/0.600;
    vrf-target target:65432:1111;
    protocols {
        l2vpn {
            encapsulation-type frame-relay;
            site Shiraz-CE {
                site-identifier 1;
                interface so-0/1/0.600 {
                    remote-site-id 4;
                }
            }
        }
    }
}
```

We see this ID value appear in the NLRI advertised to the remote PE router of Chablis:

```
user@Chardonnay> show route advertising-protocol bgp 192.168.52.1
```

```
FR-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
  192.168.32.1:2:1:3/96
*                        Self                         100        I
```

The Chablis router has a similar configuration that references a site ID of 4 for its CE router. As you would expect, this value also appears in the NLRI advertised by Chablis:

```
user@Chablis> show configuration routing-instances FR-Customer protocols
l2vpn {
    encapsulation-type frame-relay;
    site Cabernet-CE {
        site-identifier 4;
        interface so-0/1/0.600;
    }
}

user@Chablis> show route advertising-protocol bgp 192.168.32.1

FR-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
  192.168.52.1:2:4:1/96
*                        Self                         100        I
```

You may notice that the configuration of the Chardonnay router includes the `remote-site-id` command while it isn't present on the Chablis router. The use of this command is predicated by the default JUNOS software behavior for Layer 2 VPNs. Within the configuration of the customer site, each local interface connected to the CE router is listed. The PE router then makes a default assignment of local interfaces to remote site ID values. This default value assignment begins with a value of 1 and increments by 1 for each configured interface, skipping the local site ID value in the process. For the Chardonnay router, whose local site ID is 1, the `so-0/1/0.600` interface is automatically associated with a remote site of 2. Unfortunately, the Chablis router is using a site ID value of 4. This mismatch prevents the Layer 2 VPN from forming a connection and forwarding user frames across the network. When the Chardonnay router configures `remote-site-id 4` within that interface, the automatic value assignment is not used.

Once the correct site ID is determined for each remote customer site, each PE router chooses the VPN labels for receiving and sending traffic across that connection. The VPN label is the result of adding the label base advertised by the remote PE router to the site ID of the local customer connection. The label block offset advertised by the remote PE router is then subtracted from this total value to arrive at the VPN label. As a formula, we represent this calculation as: VPN label = *label-base-remote* + *local-site-id* – *label-offset-remote*. We can utilize this formula to determine the VPN labels used in our sample network.

The Chardonnay router is advertising a label base value of 800,000, a local site ID of 1, and a label offset value of 3:

```
user@Chardonnay> show route advertising-protocol bgp 192.168.52.1 detail

FR-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 192.168.32.1:2:1:3/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.32.1:2
     Label-base: 800000, range: 2, status-vector: 0x80
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111
                  Layer2-info: encaps:FRAME RELAY, control flags:2, mtu: 0
```

The relevant information from the Chablis router shows a label base value of 800,000, a local site ID of 4, and a label offset value of 1:

```
user@Chablis> show route advertising-protocol bgp 192.168.32.1 detail

FR-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 192.168.52.1:2:4:1/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.52.1:2
     Label-base: 800000, range: 1, status-vector: 0x0
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:1111
                  Layer2-info: encaps:FRAME RELAY, control flags:2, mtu: 0
```

From the perspective of Chablis, the label base advertised by Chardonnay (800,000) is added to the local site ID value of 4. This new value of 800,004 then has the label offset value advertised by Chardonnay (3) subtracted from it to arrive at a final label value of 800,001. In other words, VPN label = *label-base-remote* + *local-site-id* – *label-offset-remote* = 800,000 + 4 – 3 = 800,001.

We see this calculated value in the output of the `show l2vpn connections` command, which displays the status of each Layer 2 VPN configured on the local router:

```
user@Chablis> show l2vpn connections
Connections:
```

```
Legend for connection status (St)
OR -- out of range              WE -- intf encaps != instance encaps
EI -- encapsulation invalid     Dn -- down
EM -- encapsulation mismatch    VC-Dn -- Virtual circuit down
CM -- control-word mismatch     -> -- only outbound conn is up
CN -- circuit not present       <- -- only inbound  conn is up
OL -- no outgoing label         Up -- operational
NC -- intf encaps not CCC/TCC   XX -- unknown
NP -- interface not present

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


Instance: FR-Customer
Local site: Cabernet-CE (4)
    connection-site          Type  St     Time last up          # Up trans
    1                        rmt   Up     Nov  6 12:50:09 2001           1
      Local interface: so-0/1/0.600, Status: Up, Encapsulation: FRAME RELAY
      Remote PE: 192.168.32.1, Negotiated control-word: Yes (Null)
      Incoming label: 800000, Outgoing label: 800001
```

In addition, we see the VPN label value used by Chardonnay (800,000) for forwarding traffic to Chablis.

Once each PE router receives an NLRI from its remote peer and calculates the VPN label to use for forwarding traffic, we can verify the final end-to-end connectivity using an ICMP ping:

```
user@Shiraz> ping 10.222.100.2 rapid
PING 10.222.100.2 (10.222.100.2): 56 data bytes
!!!!!
--- 10.222.100.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.745/0.819/1.083/0.132 ms
```

ATM as the PE-CE Connection

A second Layer 2 encapsulation that can be transported across a Layer 2 VPN is ATM. One main difference occurs in this environment, however, that sets it apart from a native ATM network. When the PE routers are configured with the `atm-ccc-vc-mux` encapsulation, which is the most basic ATM type of Layer 2 VPN encapsulation, the JUNOS software reassembles the ATM cells received from the CE router and forms an AAL5 protocol data unit before transporting it across the provider network. The remote PE router then segments the PDU back into ATM cells for transmission to the remote CE router.

The sample network in Figure 9.12 is supporting an ATM connection between the Sherry and Cabernet routers. The interface configurations of the PE and CE routers are:

```
user@Sherry> show configuration interfaces at-0/1/0
atm-options {
    vpi 0 {
        maximum-vcs 1000;
    }
}
unit 800 {
    vci 0.800;
    family inet {
        address 10.222.150.1/24;
    }
}

user@Sangiovese> show configuration interfaces at-0/1/0
atm-options {
    vpi 0 {
        maximum-vcs 1000;
    }
}
unit 800 {
    encapsulation atm-ccc-vc-mux;
    vci 0.800;
}

user@Cabernet> show configuration interfaces at-0/0/0
atm-options {
    vpi 0 {
        maximum-vcs 1000;
    }
}
unit 800 {
    vci 0.800;
    family inet {
        address 10.222.150.2/24;
    }
}

user@Chablis> show configuration interfaces at-0/2/1
atm-options {
    vpi 0 {
        maximum-vcs 1000;
    }
}
unit 800 {
    encapsulation atm-ccc-vc-mux;
    vci 0.800;
}
```

The configuration of the routing instances on the PE routers shows a similar setup to what we saw with Frame Relay:

```
user@Sangiovese> show configuration routing-instances
ATM-Customer {
    instance-type l2vpn;
    interface at-0/1/0.800;
    vrf-target target:65432:2222;
    protocols {
        l2vpn {
            encapsulation-type atm-aal5;
            site Sherry-CE {
                site-identifier 1;
                interface at-0/1/0.800;
            }
        }
    }
}

user@Chablis> show configuration routing-instances ATM-Customer protocols
l2vpn {
    encapsulation-type atm-aal5;
    site Cabernet-CE {
        site-identifier 2;
        interface at-0/2/1.800;
    }
}
```

Since the provider core is already operational, we see the appropriate NLRI advertised between the PE routers:

```
user@Sangiovese> show route advertising-protocol bgp 192.168.52.1 detail
```

```
ATM-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 192.168.24.1:3:1:1/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.24.1:3
     Label-base: 800000, range: 2, status-vector: 0x80
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:2222
                  Layer2-info: encaps:ATM AAL5, control flags:2, mtu: 0


user@Chablis> show route advertising-protocol bgp 192.168.24.1 detail

ATM-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

* 192.168.52.1:4:2:1/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.52.1:4
     Label-base: 800002, range: 1, status-vector: 0x0
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:2222
                  Layer2-info: encaps:ATM AAL5, control flags:2, mtu: 0
```

The Layer 2 VPN connection is operational, and each PE router has calculated the appropriate VPN label to use for transmitting information to the remote router:

```
user@Sangiovese> show l2vpn connections
Connections:

Legend for connection status (St)
OR -- out of range              WE -- intf encaps != instance encaps
EI -- encapsulation invalid     Dn -- down
EM -- encapsulation mismatch    VC-Dn -- Virtual circuit down
CM -- control-word mismatch     -> -- only outbound conn is up
CN -- circuit not present       <- -- only inbound  conn is up
OL -- no outgoing label         Up -- operational
NC -- intf encaps not CCC/TCC   XX -- unknown
NP -- interface not present
```

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


Instance: ATM-Customer
Local site: Sherry-CE (1)
    connection-site          Type  St     Time last up          # Up trans
    2                        rmt   Up     Jul 16 14:32:58 2003           1
      Local interface: at-0/1/0.800, Status: Up, Encapsulation: ATM AAL5
      Remote PE: 192.168.52.1, Negotiated control-word: Yes (Null)
      Incoming label: 800001, Outgoing label: 800002
```

We can now verify the end-to-end connectivity from the CE routers:

```
user@Sherry> ping 10.222.150.2 rapid
PING 10.222.150.2 (10.222.150.2): 56 data bytes
!!!!!
--- 10.222.150.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.832/2.026/2.397/0.193 ms
```

Ethernet VLANs as the PE-CE Connection

Many end users are familiar with the operation and configuration of an Ethernet VLAN. Therefore, the support for transporting these Layer 2 frames across a provider network is a critical component of any technology. While the JUNOS software supports the transport of both tagged and untagged Ethernet frames, we'll focus on interfaces configured with VLAN information. Figure 9.12 displays a Layer 2 VPN employing VLAN-tagged Ethernet frames between the Sherry and Shiraz routers. The interface configurations of the respective routers are:

```
user@Sherry> show configuration interfaces fe-0/3/3
vlan-tagging;
unit 900 {
    vlan-id 900;
    family inet {
        address 10.222.200.1/24;
    }
}

user@Sangiovese> show configuration interfaces fe-0/0/3
vlan-tagging;
encapsulation vlan-ccc;
unit 900 {
    encapsulation vlan-ccc;
    vlan-id 900;
}

user@Chardonnay> show configuration interfaces fe-0/0/0
vlan-tagging;
encapsulation vlan-ccc;
unit 900 {
    encapsulation vlan-ccc;
    vlan-id 900;
}

user@Shiraz> show configuration interfaces fe-0/0/0
vlan-tagging;
unit 900 {
    vlan-id 900;
    family inet {
        address 10.222.200.2/24;
    }
}
```

WARNING

The VLAN ID must be greater than or equal to 512 for the Layer 2 VPN connections. In addition, the values must match on both sides of the provider network. This is a difference from what we saw with Frame Relay and ATM, where the Layer 2 identifier could be different on the CE routers.

The configurations of the routing instances on the PE routers look like this:

```
user@Chianti> show configuration routing-instances VLAN-Customer protocols
l2vpn {
    encapsulation-type ethernet-vlan;
    site Sherry-CE {
        site-identifier 1;
        interface fe-0/3/0.900;
    }
}

user@Chardonnay> show configuration routing-instances VLAN-Customer protocols
l2vpn {
    encapsulation-type ethernet-vlan;
    site Shiraz-CE {
        site-identifier 2;
        interface fe-0/0/0.900;
    }
}
```

Our established provider core network allows the PE routers to advertise the appropriate NLRI:

```
user@Sangiovese> show route advertising-protocol bgp 192.168.32.1 detail

VLAN-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 192.168.20.1:3:1:1/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.20.1:3
     Label-base: 800000, range: 2, status-vector: 0x80
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:3333
                  Layer2-info: encaps:VLAN, control flags:2, mtu: 0

user@Chardonnay> show route advertising-protocol bgp 192.168.24.1 detail

VLAN-Customer.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

* 192.168.32.1:4:2:1/96 (1 entry, 1 announced)
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.32.1:4
     Label-base: 800002, range: 1, status-vector: 0x0
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:3333
                  Layer2-info: encaps:VLAN, control flags:2, mtu: 0
```

Once the NLRI is received and processed by the PE routers, the appropriate VPN labels are selected and the Layer 2 VPN connection is established:

```
user@Chianti> show l2vpn connections
Connections:

Legend for connection status (St)
OR -- out of range              WE -- intf encaps != instance encaps
EI -- encapsulation invalid     Dn -- down
```

```
EM -- encapsulation mismatch    VC-Dn -- Virtual circuit down
CM -- control-word mismatch     -> -- only outbound conn is up
CN -- circuit not present       <- -- only inbound  conn is up
OL -- no outgoing label         Up -- operational
NC -- intf encaps not CCC/TCC   XX -- unknown
NP -- interface not present

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


Instance: VLAN-Customer
Local site: Sherry-CE (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Jul 17 09:44:15 2003           2
      Local interface: fe-0/3/0.900, Status: Up, Encapsulation: VLAN
      Remote PE: 192.168.32.1, Negotiated control-word: Yes (Null)
      Incoming label: 800001, Outgoing label: 800002
```

As a final step, we verify connectivity between the CE routers:

```
user@Shiraz> ping 10.222.200.1 rapid
PING 10.222.200.1 (10.222.200.1): 56 data bytes
!!!!!
--- 10.222.200.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.772/0.859/1.152/0.148
```

Using Different Encapsulations for the PE-CE Connection

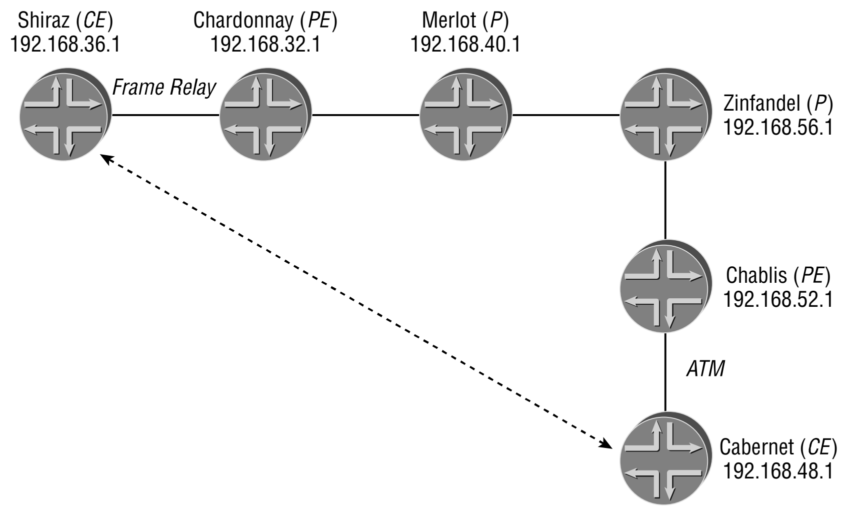
At this point in our discussion of Layer 2 VPNs, we've gained independence at the network layer by requiring the same Layer 2 encapsulation on both ends of the virtual connection. This occurs since the PE routers forward all received frames across the network without examining the contents of the network layer. A Layer 2 VPN, however, also has the ability to gain independence at the data link layer by requiring that the Layer 3 protocol be IPv4. This concept, known as *IP Interworking*, allows each PE-CE connection to use separate Layer 2 encapsulations.

A simple Layer 2 VPN using IP Interworking is shown in Figure 9.15. The connection between Shiraz and Chardonnay is using Frame Relay for its Layer 2 encapsulation, whereas ATM is used between Chablis and Cabernet. From the perspective of the CE routers, the configuration of the interfaces is no different from any other Layer 2 VPN setup:

```
user@Shiraz> show configuration interfaces so-0/1/0
encapsulation frame-relay;
```

```
unit 600 {
    dlci 600;
    family inet {
        address 10.222.222.1/24;
    }
}

user@Cabernet> show configuration interfaces at-0/0/0
atm-options {
    vpi 0 {
        maximum-vcs 1001;
    }
}
unit 0 {
    vci 0.1000;
    family inet {
        address 10.222.222.2/24;
    }
}
```

FIGURE 9.15 IP Interworking Layer 2 VPN



The knowledge of the IP Interworking connection is contained solely within the PE routers. Slight configuration changes appear in both the [edit interfaces] and the [edit routing-instances] hierarchies. The PE router of Chardonnay is configured like this:

```
user@Chardonnay> show configuration interfaces so-0/1/0
dce;
encapsulation frame-relay-tcc;
unit 0 {
    encapsulation frame-relay-tcc;
    dlci 600;
}

user@Chardonnay> show configuration routing-instances
IP-Interworking {
    instance-type l2vpn;
    interface so-0/1/0.0;
    vrf-target target:65432:4444;
    protocols {
        l2vpn {
            encapsulation-type interworking;
            site Shiraz-CE-Frame-Relay {
                site-identifier 1;
                interface so-0/1/0.0;
            }
        }
    }
}
```

On the PE interface, the encapsulation of frame-relay-tcc is configured. This allows the interface to remove the Layer 2 header from the incoming packet and forward the remaining IP information to the remote PE across the provider network. In addition, this encapsulation type provides the ability for Chardonnay to append the appropriate Layer 2 header to all received IP packets from the remote PE router.

Within the routing instance configuration, we also see some different information. In this case, the Layer 2 VPN encapsulation is set to interworking. This allows the PE routers to advertise and verify that the remote ends of the connection both support IP Interworking. We see this in the Layer 2 Information community advertised by Chardonnay:

```
user@Chardonnay> show route advertising-protocol bgp 192.168.52.1 detail

IP-Interworking.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0
hidden)
* 192.168.32.1:5:1:1/96 (1 entry, 1 announced)
```

```
 BGP group internal-peers type Internal
     Route Distinguisher: 192.168.32.1:5
     Label-base: 800000, range: 2, status-vector: 0x80
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: target:65432:4444
                  Layer2-info: encaps:INTERWORKING, control flags:0, mtu: 0
```

The remote PE router of Chablis is also configured in a similar manner to Chardonnay:

```
user@Chablis> show configuration interfaces at-0/2/1
atm-options {
    vpi 0 {
        maximum-vcs 1001;
    }
}
unit 0 {
    encapsulation atm-tcc-snap;
    vci 0.1000;
}

user@Chablis> show configuration routing-instances
IP-Interworking {
    instance-type l2vpn;
    interface at-0/2/1.0;
    vrf-target target:65432:4444;
    protocols {
        l2vpn {
            encapsulation-type interworking;
            site Cabernet-CE-ATM {
                site-identifier 2;
                interface at-0/2/1.0;
            }
        }
    }
}
```

In the routing instance configuration, we see the same `encapsulation-type interworking` statement applied. There is a special encapsulation configured on the interface as well. In this case, it is `encapsulation atm-tcc-snap`.

The presence of the appropriate VRF target communities and the Layer 2 Information community allows the Layer 2 VPN connection to become established:

```
user@Chablis> show l2vpn connections
Connections:

Legend for connection status (St)
OR -- out of range             WE -- intf encaps != instance encaps
EI -- encapsulation invalid    Dn -- down
EM -- encapsulation mismatch   VC-Dn -- Virtual circuit down
CM -- control-word mismatch    -> -- only outbound conn is up
CN -- circuit not present      <- -- only inbound  conn is up
OL -- no outgoing label        Up -- operational
NC -- intf encaps not CCC/TCC  XX -- unknown
NP -- interface not present

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


Instance: IP-Interworking
Local site: Cabernet-CE-ATM (2)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Nov 16 14:37:07 2001           1
      Local interface: at-0/2/1.0, Status: Up, Encapsulation: INTERWORKING
      Remote PE: 192.168.32.1, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800001
```

As we would now expect, the CE routers can communicate with each other. We verify this using the `ping` command from the Shiraz router:

```
user@Shiraz> ping 10.222.222.2 rapid
PING 10.222.222.2 (10.222.222.2): 56 data bytes
!!!!!
--- 10.222.222.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.330/3.182/10.028/3.425 ms
```

Layer 2 Circuit

The main difference between a Layer 2 VPN and a Layer 2 Circuit lies in the control plane and the methods used to set up the virtual connection across the provider network. The configuration of the physical interfaces as well as the actual forwarding of traffic doesn't change. In fact, both a Layer 2 VPN and a Layer 2 Circuit encapsulate the incoming customer packet in the exact same manner.

As you might expect, we spend the majority of this section discussing Layer 2 Circuit control plane issues. We follow this with a quick examination of some sample configurations using Frame Relay, ATM, and Ethernet VLAN customer connections.

Control Plane

Customer circuit information is advertised in a Layer 2 Circuit environment using the Label Distribution Protocol (LDP). The two PE routers use targeted LDP Hello messages to form a session with each other. Once the session is established, the peers exchange Forwarding Equivalence Class (FEC) information, which advertises available prefixes with an MPLS label mapping. The PE routers use this FEC advertisement to establish the virtual connection by including a new TLV that contains circuit specific information. Figure 9.16 shows the format of this TLV, which contains the following fields:

FEC Element Type (1 octet) This field displays the type of information advertised in the TLV for the FEC. A Layer 2 Circuit uses a constant value of 128 in this field.

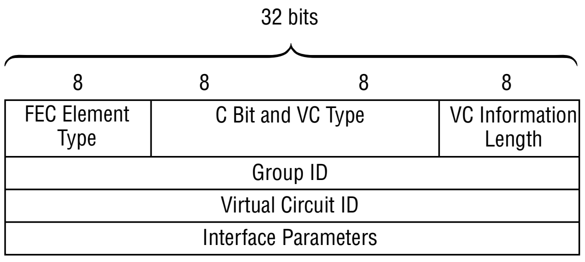
C Bit and VC Type (2 octets) The most significant bit in the field, bit 15, is called the C bit and determines whether the virtual connection should use the optional control word. When the bit is set to the value 1, the control word should be included in all packets. It should be omitted, however, when the bit is set to the value 0.

The remaining 15 bits in this field represent the type of virtual connection being established across the provider network. Possible values include:

- 0x0001—Frame Relay DLCI
- 0x0002—ATM AAL5 virtual circuit connection transport
- 0x0003—ATM transparent cell transport
- 0x0004—Ethernet virtual LAN
- 0x0005—Ethernet
- 0x0006—Cisco high-level data link control
- 0x0007—Point-to-Point Protocol
- 0x8008—Circuit emulation
- 0x0009—ATM virtual circuit connection cell transport
- 0x000a—ATM virtual path connection cell transport

VC Information Length (1 octet) This field displays the length, in octets, of the Virtual Circuit ID and Interface Parameters fields. The JUNOS software uses a constant value of 4 in this field, which means that no parameters are included in the TLV.

FIGURE 9.16 Layer 2 Circuit FEC advertisement



Group ID (4 octets) The group ID is a 32-bit value that represents a grouping of virtual circuits advertised by a PE router. This group concept is not used by the JUNOS software, and this field is set to a constant value of 0.

Virtual Circuit ID (4 octets) The *virtual circuit ID* is a 32-bit value that uniquely identifies, when combined with the VC type, a particular circuit in the network. The two PE routers must agree on the virtual circuit ID value to establish the Layer 2 Circuit in the network.

Interface Parameters (Variable) This variable-length field contains information that describes the properties of the customer-facing interface, such as the maximum transmission unit (MTU). The JUNOS software doesn't use this field, and it is not included in any FEC advertisements.

NOTE Please refer to Chapter 7, "Multiprotocol Label Switching (MPLS)," for a complete discussion on the FEC announcement message format.

Our sample network in Figure 9.12 shows multiple operational Layer 2 Circuits that connect the remote customer sites together. Each of the PE routers (Sangiovese, Chardonnay, and Chablis) has directly connected customers. These customer sites are using Frame Relay, ATM, and Ethernet VLANs to provide connectivity between themselves. Before advertising the virtual circuit information, the PE routers form targeted LDP neighbor relationships, as we see from the perspective of Chardonnay:

```
user@Chardonnay> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.168.24.1       lo0.0              192.168.24.1:0           13
192.168.52.1       lo0.0              192.168.52.1:0           11
10.222.45.1        so-0/1/1.0         192.168.40.1:0           12
```

Once established, the LDP peers advertise virtual circuit information to each other using LDP Label Mapping messages. The output of a `traceoptions` file allows us to see Chardonnay send VC information to Sangiovese:

```
Jul 28 17:24:18 LDP sent TCP PDU 192.168.32.1 -> 192.168.24.1 (none)
Jul 28 17:24:18 ver 1, pkt len 214, PDU len 210, ID 192.168.32.1:0
Jul 28 17:24:18   Msg LabelMap (0x400), len 24, ID 9506
Jul 28 17:24:18     TLV FEC (0x100), len 8
Jul 28 17:24:18       Prefix, family 1, 192.168.32.1/32
Jul 28 17:24:18     TLV Label (0x200), len 4
Jul 28 17:24:18       Label 3
Jul 28 17:24:18   Msg LabelMap (0x400), len 32, ID 9512
Jul 28 17:24:18     TLV FEC (0x100), len 16
Jul 28 17:24:18       L2CKT, VC Type 32772 VC Id 800 Group 0
Jul 28 17:24:18     TLV Label (0x200), len 4
Jul 28 17:24:18       Label 100080
```

The L2CKT notation informs us that this FEC is for a virtual connection whose virtual circuit ID is 800. The VC Type field shows a decimal value of 32772, which we convert to a hexadecimal value of 0x8004. This means that the most-significant bit is set to a value of 1, which requires the use of the control word as part of the physical encapsulation. The remaining bits in the field (0x0004) represent an Ethernet VLAN virtual connection.

In a similar fashion, Chardonnay receives a Label Mapping message from Sangiovese, which also advertises an Ethernet VLAN virtual circuit connection using an ID of 800:

```
Jul 28 17:24:18 LDP rcvd TCP PDU 192.168.24.1 -> 192.168.32.1 (none)
Jul 28 17:24:18 ver 1, pkt len 214, PDU len 210, ID 192.168.24.1:0
Jul 28 17:24:18   Msg LabelMap (0x400), len 24, ID 185177
Jul 28 17:24:18     TLV FEC (0x100), len 8
Jul 28 17:24:18       Prefix, family 1, 192.168.24.1/32
Jul 28 17:24:18     TLV Label (0x200), len 4
Jul 28 17:24:18       Label 3
Jul 28 17:24:18   Msg LabelMap (0x400), len 32, ID 185183
Jul 28 17:24:18     TLV FEC (0x100), len 16
Jul 28 17:24:18       L2CKT, VC Type 32772 VC Id 800 Group 0
Jul 28 17:24:18     TLV Label (0x200), len 4
Jul 28 17:24:18       Label 100080
```

Both PE routers now have the ability to match virtual circuit ID values, connection types, and control word parameters for this customer. This allows for the establishment of a Layer 2 Circuit between these routers using the advertised label values as the inner VPN label.

Frame Relay as the PE-CE Connection

The Shiraz and Cabernet routers in Figure 9.12 have a Frame Relay circuit established between themselves across the provider network. The CE routers connect to Chardonnay and Chablis, respectively, in the provider network. The interface configurations of the Shiraz and Chardonnay routers are:

```
user@Shiraz> show configuration interfaces so-0/1/0
encapsulation frame-relay;
```

```
unit 600 {
    dlci 600;
    family inet {
        address 10.222.100.1/24;
    }
}

user@Chardonnay> show configuration interfaces so-0/1/0
dce;
encapsulation frame-relay-ccc;
unit 600 {
    encapsulation frame-relay-ccc;
    dlci 600;
}
```

On the PE routers, the configuration of the Layer 2 Circuit occurs within the `[edit protocols l2circuit]` configuration hierarchy. The local PE associates the remote PE router with the customer interface as well as the virtual circuit ID designated for the connection. Chardonnay is configured like this:

```
user@Chardonnay> show configuration protocols l2circuit
neighbor 192.168.24.1 {
    interface fe-0/0/0.800 {
        virtual-circuit-id 800;
    }
}
neighbor 192.168.52.1 {
    interface so-0/1/0.600 {
        virtual-circuit-id 600;
    }
}
```

Because LDP is required to advertise the virtual circuit information, we also use LDP as the signaling mechanism for packet forwarding across the provider core. This allows each of the PE routers to learn what MPLS label to use for forwarding between the PE routers. This allows the Layer 2 connection to transition to the `Up` state:

```
user@Chardonnay> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface not present
MM -- mtu mismatch               Dn -- down
```

```
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
OL -- no outgoing label          XX -- unknown
NC -- intf encaps not CCC/TCC

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

Neighbor: 192.168.24.1
    Interface                 Type  St     Time last up          # Up trans
    fe-0/0/0.800 (vc 800)     rmt   Up     Jul 26 13:13:50 2003           1
      Local interface: fe-0/0/0.800, Status: Up, Encapsulation: VLAN
      Remote PE: 192.168.24.1, Negotiated control-word: Yes (Null)
      Incoming label: 100080, Outgoing label: 100080
Neighbor: 192.168.52.1
    Interface                 Type  St     Time last up          # Up trans
    so-0/1/0.600 (vc 600)     rmt   Up     Jul 26 13:16:18 2003           1
      Local interface: so-0/1/0.600, Status: Up, Encapsulation: FRAME RELAY
      Remote PE: 192.168.52.1, Negotiated control-word: Yes (Null)
      Incoming label: 100096, Outgoing label: 100096
```

Finally, we verify the end-to-end connectivity from the CE routers:

```
user@Shiraz> ping 10.222.100.2 rapid
PING 10.222.100.2 (10.222.100.2): 56 data bytes
!!!!!
--- 10.222.100.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.749/0.825/1.095/0.135 ms
```

NOTE

Just as we saw with a Layer 2 VPN, the MPLS connection between the PE routers can be established with either LDP or RSVP.

ATM as the PE-CE Connection

The Cabernet and Sherry routers are using ATM as their Layer 2 connection in Figure 9.12. These routers are physically connected to the PE routers of Chablis and Sangiovese. The current interface configurations of Cabernet and Chablis look like this:

```
user@Cabernet> show configuration interfaces at-0/0/0
atm-options {
    vpi 0 {
        maximum-vcs 1001;
    }
}
unit 700 {
    vci 0.700;
    family inet {
        address 10.222.150.2/24;
    }
}

user@Chablis> show configuration interfaces at-0/2/1
atm-options {
    vpi 0 {
        maximum-vcs 1001;
    }
}
unit 700 {
    encapsulation atm-ccc-vc-mux;
    vci 0.700;
}
```

The Layer 2 Circuit configuration of the Chablis router is currently:

```
user@Chablis> show configuration protocols l2circuit
neighbor 192.168.24.1 {
    interface at-0/2/1.700 {
        virtual-circuit-id 700;
    }
}
neighbor 192.168.32.1 {
    interface so-0/1/0.600 {
        virtual-circuit-id 600;
    }
}
```

The remote PE router of Sangiovese has a similar Layer 2 Circuit configuration, which allows the connection to become established on Chablis:

```
user@Chablis> show l2circuit connections
Layer-2 Circuit Connections:
```

```
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
OL -- no outgoing label          XX -- unknown
NC -- intf encaps not CCC/TCC

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

Neighbor: 192.168.24.1
    Interface                 Type  St     Time last up          # Up trans
    at-0/2/1.700 (vc 700)     rmt   Up     Nov 16 15:45:48 2001           1
      Local interface: at-0/2/1.700, Status: Up, Encapsulation: ATM AAL5
      Remote PE: 192.168.24.1, Negotiated control-word: Yes (Null)
      Incoming label: 100080, Outgoing label: 100096
Neighbor: 192.168.32.1
    Interface                 Type  St     Time last up          # Up trans
    so-0/1/0.600 (vc 600)     rmt   Up     Nov 16 15:45:48 2001           1
      Local interface: so-0/1/0.600, Status: Up, Encapsulation: FRAME RELAY
      Remote PE: 192.168.32.1, Negotiated control-word: Yes (Null)
      Incoming label: 100096, Outgoing label: 100096
```

Connectivity between the CE routers is also operational, which we verify with the `ping` command:

```
user@Cabernet> ping 10.222.150.1 rapid
PING 10.222.150.1 (10.222.150.1): 56 data bytes
!!!!!
--- 10.222.150.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.460/1.831/2.488/0.346 ms
```

Ethernet VLANs as the PE-CE Connection

The Sherry and Shiraz routers in Figure 9.12 are using Ethernet VLANs for their Layer 2 connection and physically connect to Sangiovese and Chardonnay. The interface configurations of the Sherry and Sangiovese routers are:

```
user@Sherry> show configuration interfaces fe-0/3/3
vlan-tagging;
unit 800 {
    vlan-id 800;
    family inet {
        address 10.222.200.1/24;
    }
}

user@Sangiovese> show configuration interfaces fe-0/0/3
vlan-tagging;
encapsulation vlan-ccc;
unit 800 {
    encapsulation vlan-ccc;
    vlan-id 800;
}
```

The Sangiovese router has a Layer 2 Circuit configuration to Chardonnay, which looks like this:

```
user@Sangiovese> show configuration protocols l2circuit
neighbor 192.168.32.1 {
    interface fe-0/0/3.800 {
        virtual-circuit-id 800;
    }
}
neighbor 192.168.52.1 {
    interface at-0/1/0.700 {
        virtual-circuit-id 700;
    }
}
```

The connection between Sangiovese and Chardonnay is now in the `Up` state:

```
user@Sangiovese> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
OL -- no outgoing label          XX -- unknown
NC -- intf encaps not CCC/TCC
```

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

Neighbor: 192.168.32.1
    Interface                 Type  St     Time last up          # Up trans
    fe-0/0/3.800 (vc 800)     rmt   Up     Jul 26 12:23:53 2003           1
      Local interface: fe-0/0/3.800, Status: Up, Encapsulation: VLAN
      Remote PE: 192.168.32.1, Negotiated control-word: Yes (Null)
      Incoming label: 100080, Outgoing label: 100080
Neighbor: 192.168.52.1
    Interface                 Type  St     Time last up          # Up trans
    at-0/1/0.700 (vc 700)     rmt   Up     Jul 26 12:26:16 2003           1
      Local interface: at-0/1/0.700, Status: Up, Encapsulation: ATM AAL5
      Remote PE: 192.168.52.1, Negotiated control-word: Yes (Null)
      Incoming label: 100096, Outgoing label: 100080
```

As we would expect, the CE routers have end-to-end connectivity:

```
user@Sherry> ping 10.222.200.2 rapid
PING 10.222.200.2 (10.222.200.2): 56 data bytes
!!!!!
--- 10.222.200.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.808/0.886/1.163/0.139 ms
```

Summary

In this chapter, we examined Layer 3 Virtual Private Networks (VPN) in some detail. We began the chapter with a discussion of the terminology used within a provider-provisioned VPN. We saw the difference between a customer edge router, a provider edge router, and a provider router. Once the basic information concerning a VPN was established, we explored how the PE routers exchange routing information with each other using multiprotocol BGP sessions to exchange VPN-IPv4 network layer reachability information. The routes are kept separate within the provider's network through the use of an 8-octet route distinguisher. In addition, the routes contain a BGP extended community value called the route target, which ensures that the correct PE routers accept the routes for the VRF tables they support. We then discussed configuration options for establishing a basic Layer 3 VPN. This included a look at using BGP or OSPF as a routing protocol between the PE and CE routers. The next topic in our exploration was a look at providing Internet access to VPN customers. This usually requires multiple logical circuits between the PE and CE routers and the advertisement of the customer's address space to the Internet at large.

We concluded the chapter with a discussion of transporting customer Layer 2 frames across the provider network. We saw that the JUNOS software supports both a Layer 2 VPN as well as a Layer 2 Circuit. The main difference between the technologies was the method used to advertise the customer virtual connection information. We used MBGP within a Layer 2 VPN, whereas the Layer 2 Circuit used LDP.

Exam Essentials

Be able to describe the basic concept of a VPN. A virtual private network (VPN) is the operation of a common physical infrastructure where each customer remains segregated within its own virtual connections. Some common examples of "traditional" VPNs include leased lines, Frame Relay, and ATM.

Be able to describe the functions of the CE, PE, and P routers. The CE router advertises routes from the customer site to its connected PE router using a Layer 3 routing protocol. The PE receives these routes and assigns both an RD and a route target to the routes. They are then advertised to remote PE routers over MBGP sessions. The routes are then readvertised to the CE routers. The P routers, on the other hand, are solely responsible for forwarding MPLS packets between the PE routers.

Be able to identify the format of a VPN-IPv4 NLRI. The routes advertised between PE routers over their MBGP sessions use a specialized format. This format includes an MPLS label allocated by the originating PE router, an RD, and the actual customer route. This combination of information allows the PE routers to keep the customer VPNs separate as well as forward traffic appropriately within each VPN.

Be able to describe the use of a route target. VPN route targets are BGP extended communities that the PE routers use to logically build the VPN for each customer. The advertising PE router assigns a route target to the customer routes before advertising them to the remote PE routers. Each receiving PE router uses its locally configured route targets to determine which inbound MBGP routes to accept. The route targets further determine which specific VRF table the received routes are placed in.

Be able to describe the advertisement of Layer 2 VPN information. Once the interface configurations and physical encapsulations are configured correctly, knowledge of this connection is advertised between PE routers. This advertisement uses MBGP to send virtual circuit data between the PE routers.

Be able to describe how a connection is established for a Layer 2 Circuit. The two PE routers on either end of the Layer 2 Circuit form an LDP neighbor relationship between themselves using targeted Hello messages. Once they further establish an LDP session, each PE router advertises the customer circuit information in a Label Mapping message.

Review Questions

1. In a Layer 3 VPN, what are the roles of the CE router? (Choose two)
   - **A.** Assign an MPLS label for the routes.
   - **B.** Forward traffic using MPLS labels.
   - **C.** Advertise routes from a site to the provider network.
   - **D.** Receive routes from the provider network about remote sites.
2. What is the correct order for the fields in a VPN-IPv4 NLRI?
   - **A.** Mask, Route Distinguisher, MPLS Label, IP Prefix
   - **B.** Mask, MPLS Label, Route Distinguisher, IP Prefix
   - **C.** MPLS Label, Route Distinguisher, Mask, IP Prefix
   - **D.** Route Distinguisher, MPLS Label, Mask, IP Prefix
3. Which JUNOS software command automatically assigns a route target of `target:64512:1234` to routes in a particular Layer 3 VRF table?
   - **A.** `vrf-import target:64512:1234`
   - **B.** `vrf-export target:64512:1234`
   - **C.** `vrf-target target:64512:1234`
   - **D.** `vrf-route-target target:64512:1234`
4. In a Layer 3 VPN environment using RSVP for MPLS reachability, how many labels are placed on a data packet by the ingress PE router?
   - **A.** 1
   - **B.** 2
   - **C.** 3
   - **D.** 4
5. When a CE and PE router are exchanging routes via EBGP sessions, which JUNOS software command on the PE router is useful for ensuring that the advertised routes are received by the CE routers?
   - **A.** `as-loops`
   - **B.** `as-override`
   - **C.** `local-as`
   - **D.** `peer-as`

6. In a Layer 3 VPN environment using OSPF between the PE and CE routers, what attribute is always advertised to the remote PE router?

   A. Route type

   B. Domain ID

   C. VPN route tag

   D. Route origin

7. Which JUNOS software configuration command allows PE routers to exchange routes in a Layer 3 VPN environment?

   A. `family inet unicast`

   B. `family inet any`

   C. `family inet-vpn unicast`

   D. `family inet-vpn vpn-nlri`

8. Which protocol is used to advertise customer circuit information for a Layer 2 Circuit?

   A. LDP

   B. IS-IS

   C. BGP

   D. MBGP

9. Which JUNOS software configuration command allows PE routers to exchange routes in a Layer 2 VPN environment?

   A. `family inet-vpn unicast`

   B. `family inet-vpn vpn-nlri`

   C. `family l2vpn unicast`

   D. `family l2vpn vpn-nlri`

10. Which Layer 2 VPN access technology requires the data-link identifier to match on both sides of the provider network?

    A. Frame Relay

    B. ATM

    C. Ethernet VLAN

    D. IP Interworking

Answers to Review Questions

1. C, D. The CE router operates a Layer 3 routing protocol and communicates with its attached PE router. It advertises and receives routes specific to its own network environment.
2. B. The VPN-IPv4 NLRI first contains a mask to advertise the number of bits contained in the NLRI. The MPLS Label and Route Distinguisher fields follow, which are used to uniquely identify the VPN routes. Finally, the IP prefix is the actual route advertised by the CE router.
3. C. The `vrf-target` command is used to automatically assign a route target to all routes in a VRF table. Both the `vrf-import` and `vrf-export` commands apply routing policies to the VRF table. Although these routing policies may assign route targets to the advertised routes, it is not an automatic functionality.
4. B. The ingress PE router adds two MPLS labels to data packets received from a local CE router. The top label value is used to reach the remote PE router, whereas the bottom label is used by the remote PE router to forward the packet to the appropriate CE router.
5. B. By default, a PE router appends its local AS to a route and advertises it to the CE router. The CE router, in turn, sees this as a looped route as its local AS is already in the AS Path. The `as-override` command allows the advertising PE router to replace the CE router's AS number with its own before the default prepend action takes place. This keeps the CE router from rejecting the route due to an AS Path loop.
6. A. The OSPF route type attribute is a BGP extended community that contains information about advertised OSPF routes. It describes the type of LSA advertisement that was received by the CE router as well as the area configured on the PE router. Receiving PE routers use this value, in conjunction with other VRF attributes, to determine if the route is advertised in a Type 3 or Type 5 LSA.
7. C. In order for two PE routers to communicate over an MBGP session and advertise NLRI for a Layer 3 VPN, the `family inet-vpn unicast` command must be applied to the BGP configuration.
8. A. While the formation of the MPLS connection between the two PE routers can be formed in multiple ways, the advertisement of customer circuit information must occur using LDP.
9. C. Two PE routers use MBGP to advertise customer circuit information when the `family l2vpn unicast` command is configured within BGP.
10. C. Only Ethernet VLANs require that the VLAN ID match on both sides of the provider network. The remaining Layer 2 access technologies of Frame Relay and ATM allow different identifiers on each side of the network.