Multiprotocol Label Switching (MPLS)

JNCIA EXAM OBJECTIVES COVERED IN THIS CHAPTER:

- ✓ **Define the functions of the following MPLS terms: LSP; LSR; ingress; transit; penultimate; egress**
- ✓ **Describe the differences between a static and a signaled LSP**
- ✓ **Define the functions of RSVP attributes**
- ✓ **Describe the relationship between BGP and MPLS**
- ✓ **Compare the differences between a strict and a loose ERO**

In this chapter, we discuss Multiprotocol Label Switching (MPLS). This topic has gained a lot of importance and popularity in the last few years. A primary reason for this growth is that MPLS is seen by many Internet engineers as a method to support multiple services over a common IP infrastructure. We won't get into the marketing and hype of what MPLS might be used for. Instead, we look at how and why MPLS was first created. Then, we discuss the basic terms and concepts, as well as the various types and methods of MPLS connections. Finally, we explore how to configure, set up, and monitor MPLS connections in the JUNOS software.

The Creation of MPLS

The use of MPLS in the Internet today is a little different from the use for which it was created. Initially, engineers designed the MPLS technology to enhance the speed and time it took a router to perform a route lookup. This speed enhancement would in turn reduce the transit delays across the Internet.

Traditional routers performed their operations in software using a central CPU architecture. Both routing protocol maintenance and traffic forwarding using a route lookup followed this pattern. As network bandwidth and capacity grew, routers of this type had a harder time maintaining efficient operations. Routing vendors felt that adapting some ATM concepts for IP routing could alleviate these bottlenecks. ATM switches performed traffic forwarding in hardware, not software. Forwarding paths were preestablished through the ATM switched network, and traffic flows were switched using a fixed-sized cell header length of 5 bytes. These concepts formed the basis for MPLS, which used a fixed-size header length and forwarded traffic based on a switching table along a predetermined path.

Before MPLS could see widespread deployment in a production environment, the landscape of the Internet had changed. Some routers were no longer performing route lookups in software. Routing vendors had used advances in silicon technology to create hardware-based application-specific integrated circuit (ASIC) routing tables. As a result, routers could now route as fast as, if not faster than, ATM switches could switch. Thus, MPLS was left without a reason to exist. Luckily, another problem arose in which a technology like MPLS could prove useful—traffic engineering.

The History of Traffic Engineering and MPLS

In a network, *traffic engineering* is the ability to control how packets get from one edge of the network to the other. Voice, LAN, and WAN networks each carry with them some form of traffic engineering. The various methods have been oftentimes crude or sometimes quite elegant. We focus on engineering methods available to ISPs and WAN networks because this is where MPLS is most often used. As we'll see, early forms of traffic engineering were very rudimentary.

IP Routing

In the dark ages of the Internet, say 1990, backbones consisted mainly of dedicated leased lines at speeds between 1.544Mbps (T1) and 44.736Mbps (T3). Each network had only a few routers and links, enabling administrators to use Interior Gateway Protocol (IGP)–based metrics to control traffic flows.

Figure 11.1 shows a sample network using this system. In this network, the router Cabernet uses the Muscat-Merlot link and the Merlot-Riesling link to reach both the Riesling and Chardonnay routers. Let's say our traffic statistics show that the Merlot-Riesling link is heavily overutilized and is causing a large transit bottleneck in the network. In an attempt to resolve this issue and engineer the traffic flow, we change the IGP metric of the Merlot-Riesling link to 6, as shown in Figure 11.2.

FIGURE 11.1 IGP-based network

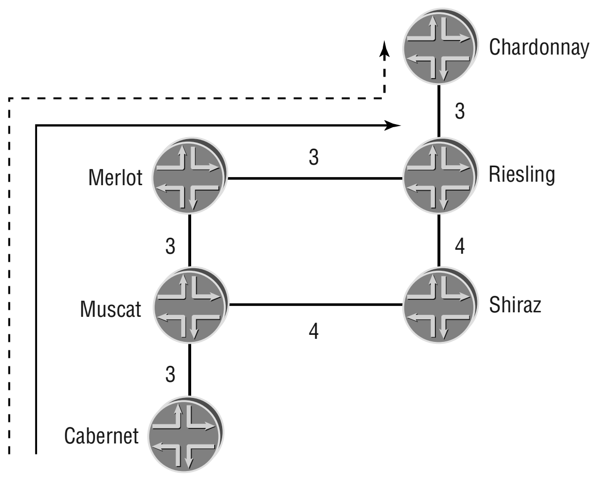
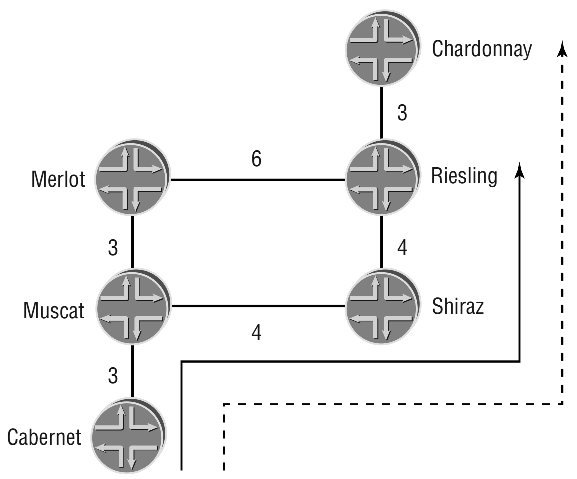


FIGURE 11.2 IGP-based traffic engineering

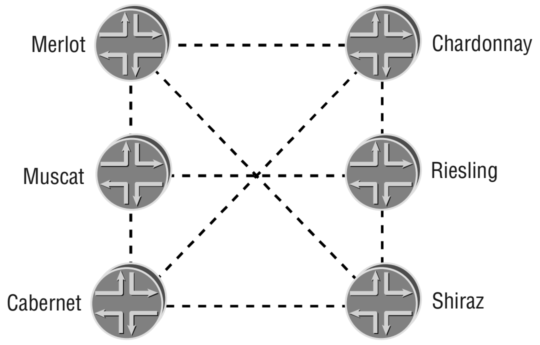


We've successfully solved the utilization problem on the Merlot-Riesling link. However, we've also created another problem. All traffic destined for both Riesling and Chardonnay now uses the Muscat-Shiraz and Shiraz-Riesling links. All we've really accomplished is moving the bottleneck to another set of links. Clearly, using IGP metrics for traffic engineering doesn't provide for fine-grained control. We need a better method. One option, ATM networks, is discussed next.

ATM and Overlay Networks

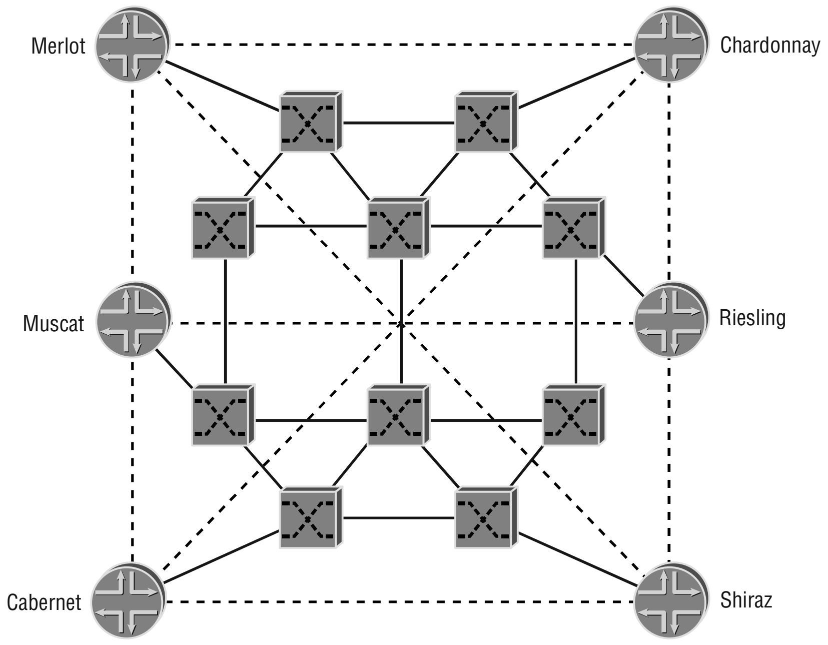
During the mid-1990s, ISPs saw traffic levels across their backbones rise dramatically. The Internet was growing in popularity, and more users and websites meant more traffic. The IGP-routed core didn't provide enough interface speed or deterministic control to face these new traffic patterns. Most ISPs turned to ATM as a core technology. Interface speeds for ATM networks started at OC-3 (155.52Mbps) and grew to OC-12 (622.08Mbps). In addition, the ATM core used virtual circuits (VCs) to logically connect the routers, as shown in Figure 11.3.

FIGURE 11.3 ATM overlay network–logical view



Each VC connection in the network appears as a point-to-point interface from the perspective of the router. In reality, however, the physical ATM connectivity may be quite different. Figure 11.4 shows this second set of connections.

FIGURE 11.4 ATM overlay network–physical view



This environment was commonly called an *overlay network*, since multiple distinct networks were operating in parallel. (In our example, the ATM network routes packets and cells at Layer 2, while the IP network routes packets between distinct VCs at Layer 3.) From a traffic engineering perspective, an overlay network provided for better control because the physical path was determined by the VC setup and not by the IGP metrics between the routers. When the traffic statistics revealed that a physical link was overutilized in an overlay network, the ATM VC connection was moved to another path. The IP network and its routers, however, did not see this change and happily routed packets over the VC using the IGP metrics.

Overlay networks also provided other benefits to the ISPs. One such benefit was the ability to gather statistics on a per-VC basis. In an IGP-routed core, the statistics on a physical link showed only the total amount of traffic across that connection. You didn't know who the recipients of the packets were or what types of traffic were using the link. When ATM VCs were introduced into this picture, the statistics on that physical media had some segmentation to them. Each VC on the link connected two routers, and traffic flows between those routers were now visible.

Of course, the overlay network solution had its drawbacks as well. Each of the networks (ATM and IP) required engineers and support staff who specialized in their operation, placing a drain on company resources. The separation of knowledge in the overall network also meant that the switches and routers couldn't share responsibility for engineering traffic flows. Finally, there was the issue of the ATM *cell tax*. Each 53-byte ATM cell forwarded through the network carried with it a 5-byte header and 48 bytes of payload. The transmission of the ATM headers added up to a significant amount of bandwidth. For example, it took two cells to send a 64-byte IP packet across the network. The total bandwidth used by these cells was 106 bytes. The extra 42 bytes of used transmission capacity was not beneficial to the network as a whole and represented almost 40 percent of wasted bandwidth. While this is an extreme case, the ATM cell tax averaged between 10 and 20 percent.

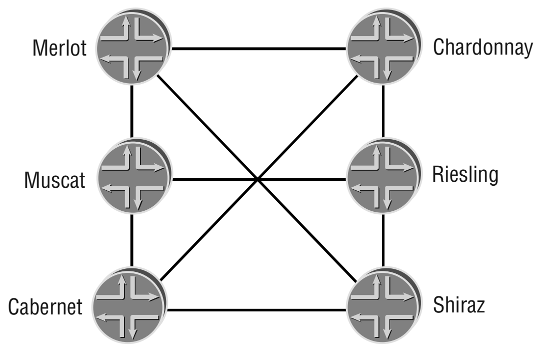
ISPs were willing to live with the drawbacks of the overlay model as long as the disadvantages were outweighed by the benefits, namely the higher interface speeds. This benefit started to dwindle at the end of the 1990s as backbone traffic increased at a steady rate. ATM interface speeds remained at OC-12 capacity with no increase in sight. The main barrier to a faster interface was the hardware responsible for the ATM segmentation and reassembly (SAR) process. ATM vendors found it very difficult to produce OC-48 and OC-192 SAR hardware at an affordable price. Consequently, neither of these speeds was ever brought to market in large quantities. Faster speeds were needed to handle the increased traffic flows.

SONET and MPLS

As the end of the 1990s approached, router vendors were able to produce interfaces that operated with the Synchronous Optical Network (SONET) specification. This allowed interface and backbone speeds to increase to OC-48/STM-16 (2488.32Mbps/2.5Gbps) and OC-192/STM-48 (9953.28Mbps/10Gbps) capacity. While ISPs were happy with this development, they wanted to enjoy the benefits of the ATM overlay network model as well. This became the function of MPLS.

In an MPLS-based network, only IP knowledgeable devices exist to route traffic across the network. The routers are connected with point-to-point WAN interfaces running an IGP. This type of model is seen in Figure 11.5.

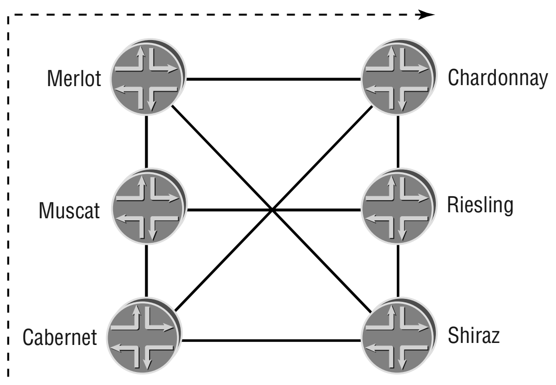
FIGURE 11.5 MPLS and IP–based network



At first glance, this looks like a regular IGP-routed core with faster interfaces. However, the addition of MPLS as an operating protocol greatly alters the operation of the network. Each router now has the ability to create a label switched path (LSP) to any other router in the network. This path can traverse any number of physical links in the network.

Figure 11.6 shows an MPLS path from Cabernet to Chardonnay. This new network path physically uses the Cabernet-Muscat, Muscat-Merlot, and Merlot-Chardonnay links instead of the directly connected Cabernet-Chardonnay link. Like an ATM VC, this is a logical connection that provides connectivity between two routers. It is able to move to other physical links should traffic statistics reveal a bottleneck in the network. It also has the ability to provide statistics on a per-path basis. These are all benefits that ISPs received from the overlay network model.

FIGURE 11.6 MPLS network path



An MPLS-based network also mitigates many of the disadvantages of an overlay network. The ability to use any physical transmission media allows higher backbone and interface capacity. In addition, the ATM cell tax is eliminated. The removal of ATM also reduces the network support costs—all traffic now uses the IP-based network.

The use of MPLS as a forwarding mechanism provides ISPs with an even finer-grained method for traffic engineering than they received from ATM. The traffic using an MPLS network path is now a single IP subnet. Let's contrast this to an ATM VC in the overlay network. Multiple logical VCs were able to use a single physical link, and statistics from those connections showed traffic flows between two individual routers. In an MPLS network, multiple paths can exist between those two routers, with individual IP subnets using different paths. Traffic flows between the routers are now visible to the ISP on a per-destination basis.

Now that we've examined why MPLS is being used today, let's explore how it actually works.

MPLS Operations

MPLS is still an evolving protocol, but it is supported by the vast majority of routing vendors. As a new creation, it has its own set of terms and operational parameters to use for proper operation. In this section, we discuss the MPLS terminology and then look at methods for establishing MPLS network paths.

Multiprotocol Label Switching Standards

The JUNOS software currently supports the following RFCs and Internet drafts:

- RFC 2702, "Requirements for Traffic Engineering over MPLS"
- RFC 3031, "Multiprotocol Label Switching Architecture"
- RFC 3032, "MPLS Label Stack Encoding"
- Internet draft draft-ietf-isis-traffic-02.txt, "IS-IS Extensions for Traffic Engineering"
- Internet draft draft-katz-yeung-ospf-traffic-04.txt, "Traffic Engineering Extensions to OSPF"
- Internet draft draft-ietf-mpls-icmp-02.txt, "ICMP Extensions for Multiprotocol Label Switching"

Internet drafts are limited in scope and are updated on a frequent basis. Please check `www.ietf.org` for the current names and status of listed drafts.

Terminology

The path created in an MPLS network is called a label switched path. Each MPLS enabled router in the network is considered a label switching router. Finally, the actual forwarding of packets is accomplished using a header value that contains a numeric label value. Let's take a closer look at what these terms actually mean.

Label Switched Path (LSP)

Each network path created by the MPLS protocol is a *label switched path (LSP)*. This path is a unidirectional entity that typically exists within a single autonomous system or domain. This one-way traffic flow is different from that of many ATM VCs, which are usually established in a bidirectional manner. The use of a unidirectional system allows you ultimate control of your traffic but does require LSPs to be established in both the transmit and receive directions for total traffic engineering in the network.

Real World Scenario

Are We Running Out of Acronyms?

So, an MPLS network path is called a label switched path (LSP). You may recall from Chapter 7, "IS-IS," that the acronym LSP also means a link-state PDU. Clearly, the networking world has run out of usable acronyms. We first started using acronyms from other industries. ATM is both Asynchronous Transfer Mode as well as an Automatic Teller Machine (where you get money from the bank). Now we need to reuse our own internal acronyms. The Internet is about to collapse!

Well, that's not actually true, but it makes a great story for non-networking people. In reality, we haven't run out of acronyms. We've just reused one for a good cause!

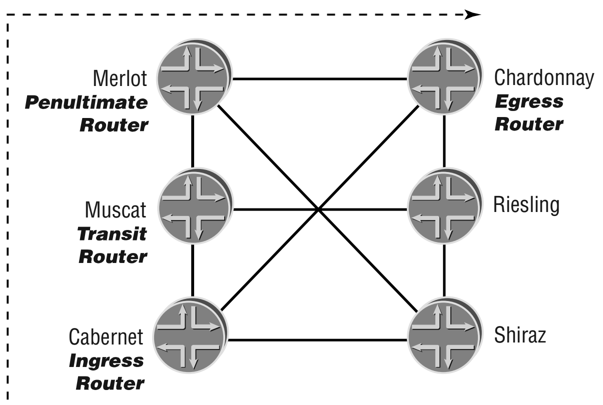
The point of all this bantering is to make sure that you are certain of your surroundings before using the term LSP. In a mixed crowd, some people may hear *link-state PDU* while others might hear *label switched path*. Within the course of this chapter, LSP means a label switched path.

Label Switching Routers (LSR)

Each IP router that supports the MPLS protocol is called a *label switching router (LSR)*. An LSR understands the MPLS header and the values encoded within it. The LSR is also responsible for the actual forwarding of user data traffic through the established LSP.

There are four different types of LSRs: ingress, transit, penultimate, and egress. Figure 11.7 shows an established LSP with each type of LSR displayed.

FIGURE 11.7 MPLS router types



The unidirectional LSP is set up from Cabernet to Chardonnay. As such, Cabernet is the ingress router and Chardonnay is the egress router. All other routers within the LSP are considered transit routers. In Figure 11.7, both Muscat and Merlot are transit routers. However, Merlot is also the last router prior to the egress router, making it the penultimate router. Each LSR type performs specific functions within the LSP operation. Let's examine these a little closer.

Ingress Router

The *ingress router* in an LSP is the only entry point for user data traffic into MPLS. Native IPv4 packets are encapsulated into the MPLS protocol at this location by way of a *label push operation*. Once encapsulated, packets flow to the egress of the LSP in a downstream fashion. Hence, the ingress router is upstream from the perspective of the data flow.

Each LSP in a network must have an ingress router. In addition, only a single ingress router may exist per LSP.

Transit Router

All routers located in the middle of an LSP are considered *transit routers*. An individual path can contain between 0 and 253 such routers. In the sample LSP in Figure 11.7, two transit routers exist, Muscat and Merlot. Should an LSP be configured between Cabernet and Shiraz, no transit routers would exist.

NOTE The upper limit of transit routers in an LSP is a function of the 8-bit TTL field of the MPLS label, with a maximum value of 255. Both the ingress and egress routers belong to the LSP, leaving 253 other possible hops along the path.

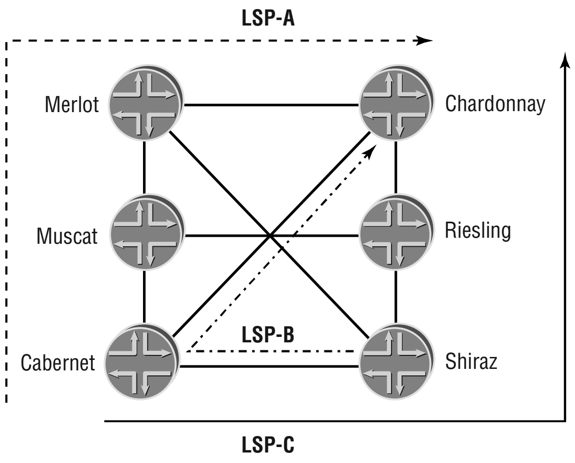
The function of a transit router is quite simple. The router checks all received MPLS packets for an incoming label value, which it then looks up in an MPLS forwarding table. After locating the label, the transit router performs a *label swap operation* by replacing the incoming label with an outgoing label value and decrements the MPLS TTL by 1. The router then forwards the newly labeled data packet to the next hop of the LSP. This entire operation never utilizes the information in the IP data header.

Penultimate Router

One of the transit routers in an LSP—the *penultimate router*—has a special function to perform. This router, which is second to last along the path of the LSP, often performs a *label pop operation* to remove the MPLS information from the data packet. After consulting the MPLS switching table, the router forwards the resulting data, a native IPv4 packet, to the next hop in the LSP after decrementing the TTL value by 1.

Performing this de-encapsulation function on the penultimate router results in scalability. Figure 11.8 shows a network with multiple LSPs established. Three of the LSPs end at Chardonnay. Each of the LSPs has a different penultimate router along the path. If Chardonnay is responsible for de-encapsulating all MPLS packets from the three LSPs, it performs a certain amount of work. This workload increases as the number of LSPs ending on Chardonnay increases. Imagine if 50 LSPs terminate there, or 100, or even 1000. The effort exerted by Chardonnay increases dramatically. If we move the de-encapsulation function to the penultimate router, however, the workload of the label pop operation is spread across a greater number of routers. This *penultimate hop popping (PHP)* system allows an MPLS network to scale to greater proportions.

FIGURE 11.8 Benefits of the penultimate router



A Juniper Networks router performs penultimate hop popping (PHP) by default on all dynamic LSPs.

Egress Router

The *egress router* is the end point of the LSP. The egress router receives packets from the penultimate router and performs an IPv4 route lookup operation. The router then forwards the data packet to the next hop of the route. From a directionality perspective, the egress router is downstream to all other routers in the LSP.

Each LSP in the network must have an egress router. As is the case with its ingress partner, only a single egress router may exist per LSP.

Popping the MPLS Label

Our definition of each router's role along the path of an LSP assumes the default JUNOS software behavior of penultimate hop popping (PHP). In this case, the penultimate transit router in the LSP performs the label pop operation. Another option exists for popping the MPLS label from the data packet—ultimate hop popping.

Ultimate hop popping occurs when the egress router itself performs the label pop operation. This requires the egress router to perform two operations on the data packet: the label pop and an IPv4 lookup to forward the packet. This dual operation places a larger processing burden on the egress router, which prompts the use of PHP.

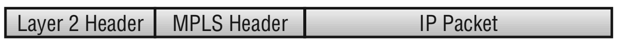
The technical difference between PHP and ultimate hop popping comes in the action of the egress router. Based on its current configuration, the egress router signals different MPLS labels upstream to the penultimate router. A label value of 3 means the upstream router should perform PHP and forward native IPv4 packets. A label value of 0, on the other hand, tells the upstream router to perform a label-swap operation and to forward the data with an MPLS header attached. (We discuss the definitions of the MPLS label values in the "Labels" section of this chapter.) A Juniper Networks router performs ultimate hop popping when the `explicit-null` command is applied.

Labels

The forwarding of user data traffic through an MPLS network is accomplished by *label values* assigned by the MPLS routers themselves. This assignment occurs in an upstream direction through a manual or dynamic process. The downstream router, in essence, informs the upstream router what label value to use when sending traffic along the LSP. When the downstream router receives that label value, it swaps the label with the value assigned by *its* downstream router. This exchange of labels between two routers on a single link results in the label value having *local significance* only. This means that a specific value, say 100101, may appear on multiple links in a network simultaneously. This is a similar concept to both ATM and Frame Relay networks and provides for excellent scalability of the network.

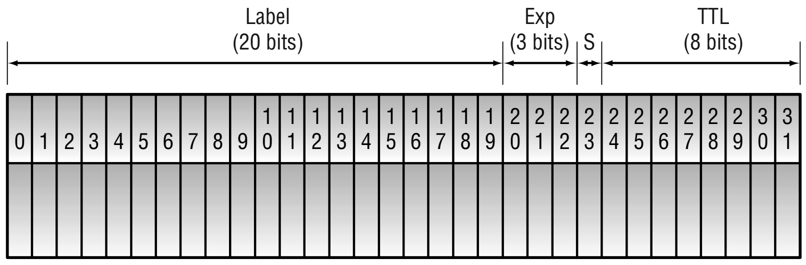
The assigned labels are encoded as part of a 32-bit MPLS *shim header* that the ingress router adds to the packet. The router places this header between the IP packet and the appropriate Layer 2 header for the physical link, as shown in Figure 11.9.

FIGURE 11.9 MPLS shim header



The format of the MPLS header is shown in Figure 11.10 and consists of the following fields:

FIGURE 11.10 MPLS header details



Label (20 bits) This field contains a locally significant value specifying that a packet belongs to a certain LSP. Possible values range from 0 to 1,048,576.

Experimental Bits (3 bits) Listed as experimental due to standards work, this field was always intended for use as a Class of Service (CoS) field. The particular type of CoS is still undetermined, hence the experimental title.

Stacking Bit (1 bit) This field indicates whether an IP packet or another MPLS header follows the current header. A value of 1 represents an IP packet, and a value of 0 means other MPLS headers follow.

Time to Live (8 bits) The same as the TTL field in an IP header, this prevents looping MPLS packets in the network. Each router decrements this field by 1, and any value of 0 results in a dropped packet. The default action for an LSP is to copy the IP TTL value to the header at ingress and copy the MPLS TTL value back to the IP packet when the label is popped.

The Internet Engineering Task Force (IETF) reserves some label values for standardized use on all MPLS routers. These values are in the range of 0 through 15. Their meanings, as defined by the IETF, are:

0 – IPv4 Explicit NULL This entry is valid only as a label when an IPv4 packet follows the MPLS header. It indicates that the label must be popped and a route lookup performed for packet forwarding.

1 – Router Alert Label This entry indicates that the packet should be sent to the Routing Engine for processing; the packet should not be forwarded based on the incoming label value.

2 – IPv6 Explicit NULL This entry is valid only as a label when an IPv6 packet follows the MPLS header. It indicates that the label must be popped and a route lookup performed for packet forwarding.

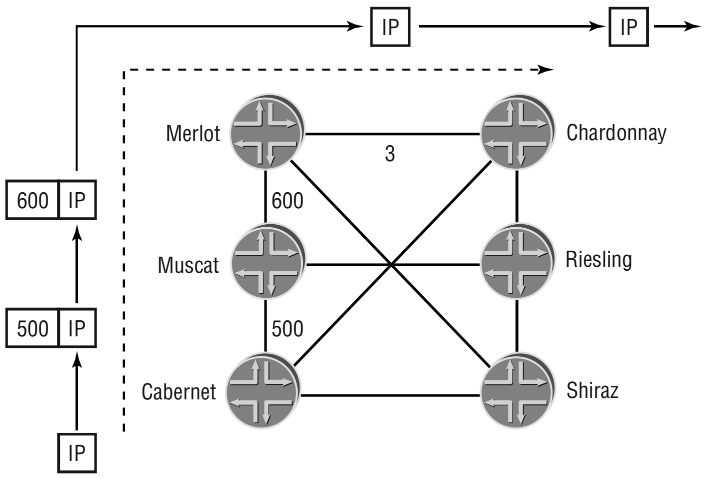
3 – Implicit NULL This label value should never appear in an MPLS header. A router receiving this value from its downstream neighbor should pop the label of all received MPLS packets and forward the remaining data to the downstream router using the information in the local MPLS switching table. This action is performed instead of performing a swap operation on the local router. This label value is used for penultimate hop popping.

4 – 15 These label values are reserved for future use.

Packet Processing

Now that we have the basic terminology under our belt, let's examine how a data packet is actually forwarded through an MPLS network. Figure 11.11 shows a sample network with an LSP and some assigned labels.

FIGURE 11.11 MPLS packet processing



As you can see, an LSP has been established from Cabernet to Chardonnay. All downstream routers have assigned label values to their upstream neighbors; Chardonnay has assigned 3, Merlot 600, and Muscat 500. When an IP packet arrives at the ingress router of Cabernet, the following sequence occurs:

1. Cabernet performs an IPv4 route lookup on the destination IP address. It finds the next hop for the route is the LSP to Chardonnay. An MPLS header is added to the packet with a label value of 500 and forwarded to Muscat.
2. Muscat receives an MPLS packet with a label of 500. It performs an MPLS forwarding table lookup and finds a swap operation. It removes the label of 500 and replaces it with a label of 600. The packet is forwarded to the next hop along the LSP.
3. Merlot receives an MPLS packet with a label of 600. It performs an MPLS forwarding table lookup and finds a pop operation since its downstream peer advertised a value of 3. Merlot removes the MPLS header from the packet and forwards the remaining data (IPv4 packet) to the next hop of the LSP.
4. Chardonnay receives an IPv4 packet. It performs a routing table lookup and forwards the packet to the next hop of the route.

One of the keys to a successful packet transmission is correctly assigned labels. This occurs as part of the establishment of the LSP itself, so let's now discuss those various methods.

Establishing an LSP

An MPLS label switch path is established by one of two methods: static or dynamic. Each method carries with it both advantages and disadvantages, as we see in the following sections.

Static Label Switched Paths

A *static label switched path* requires that each router along the LSP be configured explicitly. This is very similar to static IPv4 routes, where each hop along the path requires a static route.

One benefit to establishing a static LSP is the simple and straightforward manner of its operation. You decide where the LSP should go and what labels it should use, and you assign those resources. Static LSPs also consume fewer router resources than dynamic LSPs do. No signaling protocol is operated, and no state information is required to be maintained. Again, this is similar to static routes, which require fewer resources than routes that use a routing protocol.

Of course, the same reasons for not using static routes in a network also apply to static LSPs. The lack of knowledge about topology changes means that traffic may be black-holed during an outage. Static LSPs are active until you change their parameters in the configuration, and the process of making those changes on multiple routers is prone to error.

Dynamic Label Switched Paths

To achieve visibility into the LSP, use a *dynamic label switched path*. As the name implies, a signaling protocol creates and maintains an LSP with no user intervention. Only the ingress router is configured with the information concerning the LSP; all other routers receive signaling messages during the establishment process.

Your ability to control the setup parameters of the LSP depends primarily on your choice of a signaling protocol. The JUNOS software supports two methods of signaling an LSP, which is our next topic of discussion.

ATM VC versus MPLS LSP

As you read the "The History of Traffic Engineering and MPLS" section earlier in this chapter, you saw the very strong relationship between ATM and MPLS. Partly due to history and partly due to a similar switching paradigm, the concepts are closely aligned. As such, it might be useful to draw some correlations between a virtual circuit (VC) and a label switched path (LSP).

Static LSPs are very similar to ATM permanent virtual circuits (PVCs). Both paths are manually created along each hop and remain *nailed up* until a configuration change alters the path. There is absolutely no flexibility or contingency to the establishment and maintenance of the path. Should a node along the path stop functioning, the entire path fails to function. No fail-over capabilities are in place.

Dynamic LSPs, on the other hand, are very similar to ATM switched virtual circuits (SVCs). Both use a signaling protocol to establish the path, and configuration is needed only at the head-end node. Furthermore, a link or node problem along the path does not bring the path down. The signaling protocol establishes a new path through the network, provided an alternate path exists.

Signaling Protocols

The two signaling protocols supported in the JUNOS software are the Resource Reservation Protocol (RSVP) and the Label Distribution Protocol (LDP). RSVP is a generic signaling protocol that has been adapted for use in MPLS. LDP, on the other hand, was designed explicitly for use with MPLS. The two protocols are fully independent from each other but can be used at the same time in a network.

Resource Reservation Protocol

The IETF designed the *Resource Reservation Protocol (RSVP)* as a method for allowing end hosts to reserve capacity in a network. The theory is that applications required a certain quality of network service standards to operate correctly. RSVP is a method aimed at accomplishing this goal. Widespread use of RSVP for its original purpose never occurred for a number of reasons, one being that ISPs didn't want their customers altering the network configuration. The protocol has been extended to support traffic engineering capabilities and is currently used to establish MPLS LSPs.

Some portions of the original specification are still used, so we cover those first. We then explore more deeply the extensions to the protocol for MPLS use.

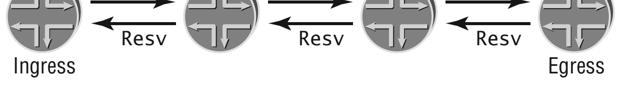
RSVP Basics

RSVP uses unidirectional and simplex (one-way) flows through the network to perform its function. The ingress router initiates an RSVP *Path message* and sends it downstream to the egress router. This `Path` message contains information about the requested resources of the connection. Each router along the path begins to maintain a *soft state* connection for this reservation. You can think of the soft state as a database of current reservations affecting the local router.

When the `Path` message reaches the egress router, the actual reservation of resources begins. This happens with an *RSVP Resv message*, which is initiated by the egress router and sent upstream to the ingress router. Each router along the path receives the `Resv` message and sends it upstream, following the route used by the `Path` message. In addition, more soft state information is added to each local router. Once the ingress router receives the `Resv` message that matches its original `Path` message, the unidirectional network path is established.

The established network path remains operational as long as the RSVP soft state stays active. This is accomplished through a refresh mechanism where each local router sends `Path` and `Resv` messages to its neighbors for all current states every 30 seconds. This informs those neighbors of active paths and assists them in maintaining their own local soft state. The flow of `Path` and `Resv` messages in a network is seen in Figure 11.12.

FIGURE 11.12 RSVP `Path` and `Resv` messages



In addition to the Path and Resv messages, RSVP defines these message types:

PathTear message The *PathTear message* always travels downstream to the egress router. It removes the established Path soft state for all routers receiving the message. A transit node sends this message when an outage occurs. The ingress router may also use it when the path is no longer desired.

ResvTear message The *ResvTear message* always travels upstream to the ingress router. It removes the established Resv soft state for all routers receiving the message. A transit node sends this message when an outage occurs.

PathErr message The *PathErr message* always travels upstream to the ingress router. It denotes an error along the established path. No soft state is removed by routers receiving this message type.

ResvErr message The *ResvErr message* always travels downstream to the egress router. It denotes an error along the established path. No soft state is removed by routers receiving this message type.

ResvConf message The egress router may ask each node along the path for a confirmation that the Resv message was received. The *ResvConf message* type provides that confirmation message.

RSVP Extensions

The RSVP extensions to support MPLS LSPs allow label-specific information to be encoded in the Path and Resv messages. In addition, path maintenance and scalability issues are addressed.

The soft state information described in the previous "RSVP Basics" section remains active on a local router for approximately three minutes. Many ISPs desire a quicker response to network changes and outages than the soft state allows. To combat this issue, extended RSVP uses a *hello mechanism*. Each RSVP router sends a hello message to its neighbors every 9 seconds, by default. When a router stops sending hello messages, its neighbors detect the change in 63 seconds and advertise the appropriate error messages. This extension is backward compatible with the original RSVP specification. Should a neighbor not support the hello mechanism, the soft state timers are used.

When each router sends Path and Resv messages to refresh the soft state database, it does so for each established path in the database. As the number of paths grows, the local router sends additional messages to its neighbors. For scalability, RSVP now supports *message aggregation*, which allows a router to bundle up to 30 messages into a single packet before sending it to a neighbor. Path, Resv, hello, and error messages are suitable for this aggregation function.

A primary goal of extending RSVP is to support MPLS LSPs. As such, some of the additions to the protocol specifically account for the establishment and maintenance of traffic-engineered LSPs. The extensions include a number of objects that are encoded within the Path and Resv messages.

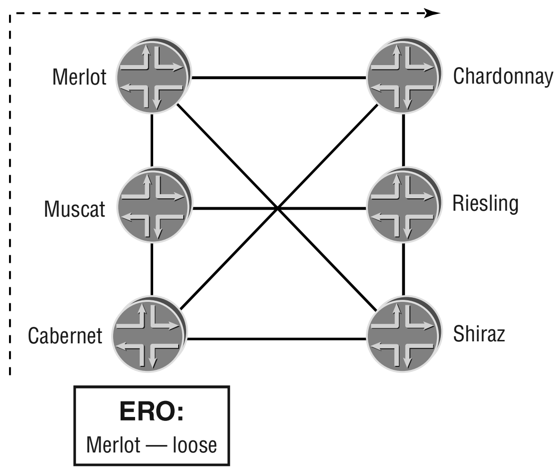
EXPLICIT ROUTE OBJECT

The *explicit route object (ERO)* allows the Path message to traverse the network using information that is independent of the IGP shortest path. The ingress router adds the ERO to the Path message, and all transit routers must follow the specifications outlined in the object.

A configured ERO may contain only *loose hops*. This specifies that the LSP must transit the specified nodes in the object in the order given. The IGP shortest path is used between the loose hop nodes.

Figure 11.13 shows an LSP using a loose hop ERO. Here, the only specified transit router in the ERO is Merlot, with an attribute of loose. The ingress router (Cabernet) and all other transit routers use the IGP routing table to send the Path message toward Merlot. Merlot examines the ERO and finds no other specified nodes. It then forwards the Path message toward the egress router using its IGP routing table.

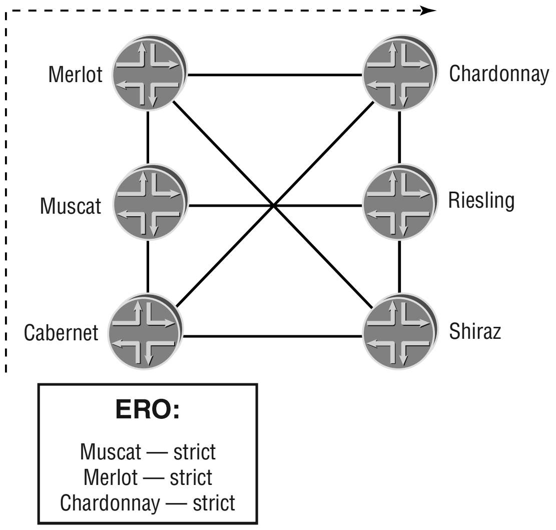
FIGURE 11.13 ERO using loose hops



The opposite of a loose hop in an ERO is a *strict hop*. When you use a strict hop, you are informing the network of the exact path you wish the LSP to take. Each router examines the defined ERO and forwards the Path message to the next router listed. The next-hop router must be directly attached to the local router for this process to succeed.

Figure 11.14 details an LSP established using a strict hop ERO. Each hop along the LSP path is specified in the ERO. The ingress router determines if Muscat is directly connected to itself. This is the case in our network, so the Path message is forwarded to Muscat. Muscat performs the same function. It finds Merlot is directly attached and forwards the Path message appropriately. Merlot examines the ERO, determines that Chardonnay is directly attached, and forwards the Path message. Chardonnay receives the Path message and notices that no more nodes are listed in the ERO and that the egress router address equals a local interface address. It then terminates the Path message and generates a Resv message back along the path.

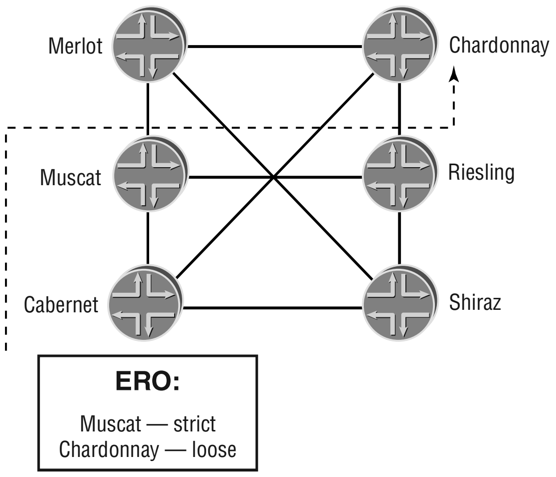
FIGURE 11.14 ERO using strict hops



WARNING Failure to have an ERO strict hop directly connected results in an error message. The requested LSP is not established in this condition.

Some situations might call for an ERO that contains both loose and strict hops. This is a perfectly allowable condition that extended RSVP can handle. The requested LSP becomes established when each router has a route to the loose hops and all strict hops are directly connected. Figure 11.15 shows an example of an ERO using both forms of next hops.

FIGURE 11.15 ERO using both loose and strict hops



In this case, only a single strict hop is configured. Cabernet, the ingress router, finds that Muscat is directly connected and forwards the Path message to the next hop. Muscat examines the ERO and finds that the next hop is loosely defined. It then uses its IGP routing table to forward the Path message. In this example, it arrives at Riesling. Riesling performs a routing lookup to find the loose hop and forwards the Path message to Chardonnay.

LABEL REQUEST OBJECT

The *label request object* is encoded in the Path messages sent to the egress router. This object allows each router to assign a label value to the requested LSP. When the Path message is received, the local router allocates a label and stores it with the Path soft state for that LSP. When the Resv message arrives from the downstream neighbor, the label is advertised upstream in an RSVP label object.

LABEL OBJECT

The *label object* is carried within the Resv messages sent to the ingress router. The object allows each router along the path to advertise its assigned label value to each upstream router.

RECORD ROUTE OBJECT

The *record route object (RRO)* can be encoded in both a Path and a Resv message. The primary purpose of the RRO is loop detection. A router does not forward an RSVP message if the current RRO lists an interface belonging to the local router.

The ingress router generates the Path message and includes the RRO. As the Path message moves downstream, each router along the path adds its outgoing interface address to the RRO. Should a loop be encountered, the Path message is dropped. A PathErr message is generated by that node and sent to the ingress reporting a "routing problem, loop detected" error.

The process works the same in reverse, with the egress router generating a Resv message with an RRO. As the Resv message moves upstream, each router along the path adds its outgoing interface address to the object. If a loop is found, the Resv message is dropped. A ResvErr message is generated by that node and sent to the egress reporting a "routing problem, loop detected" error.

SESSION ATTRIBUTE OBJECT

The *session attribute object* is contained within a Path message. MPLS routers use this object to control the priority, preemption, affinity class, and local-rerouting capabilities of the LSP. In addition, the ingress router may place an ASCII string in the session attribute object. This string assists users in identifying an LSP on each router in the network.

TSPEC OBJECT

The *tspec object* (or traffic specifier object)is also encoded within an RSVP Path message. It contains information such as the requested bandwidth of the LSP. Each MPLS router uses this data to determine whether the LSP should become established. The tspec object also contains the minimum and maximum packet sizes supported along the path of the LSP.

Label Distribution Protocol

The *Label Distribution Protocol (LDP)* is the second method for signaling the establishment of an LSP. Unlike RSVP, LDP is a new protocol designed specifically for use with MPLS.

LDP detects neighbors using a Hello protocol. Once a neighbor is found, an LDP router establishes a TCP session with that peer to exchange label information. Each set of established LDP peers generates a set of labels for inclusion in the LDP database, which is advertised throughout the network. Every LDP router then consults its database to determine the label required to reach every other LDP router in the network.

Real World Scenario

Policing Traffic in MPLS

When learning about the capabilities of the MPLS signaling protocols, many users focus on the bandwidth request capabilities of RSVP. It is very important to know what the limitations and functions of this request entails.

We've been happily drawing similarities between ATM and MPLS throughout this chapter. This is one case where this comparison breaks down. Bandwidth requests in an ATM network mean that each node along the path actively monitors the traffic flows. Should the amount of traffic exceed the requested amount, the excess traffic is dropped from the network. In short, ATM nodes perform policing functions.

An MPLS bandwidth request, however, does no such thing. No policing functions are performed within the MPLS routers. The purpose of the request is only to determine whether the LSP should be established. There is no inherent guarantee that the requested bandwidth is available in the network.

This is not to say that you can't do policing in an MPLS network. In fact, you can. However, this function must be accomplished outside the MPLS and RSVP protocols. On a Juniper Networks router, policing functions are part of the firewall filter syntax (see Chapter 10, "Firewall Filters").

MPLS Implementation

Now that we've covered the theory of MPLS, let's discuss how the protocol is implemented within the JUNOS software. We first explore how to establish an LSP using a static configuration. Next, we look at how RSVP is used for dynamic signaling. Finally, we discuss how user traffic actually enters and uses the LSP for traffic forwarding.

Configuring a Static LSP

In the “Establishing an LSP” section earlier in this chapter, we stated that each router must be configured to support a static LSP. While this is a good general statement to make, some more detail is needed. In fact, each router requires the ability to support the MPLS protocol as well as interpret MPLS packets received on an interface. When it comes to the LSP itself, the ingress and all transit routers must be correctly configured.

Let’s explore the steps required to configure static LSPs within the JUNOS software. We first configure `family mpls` on all the router interfaces. We then enable the MPLS protocol on the router and configure the routers themselves for the static LSP.

Configuring Interfaces

An interface on a Juniper Networks router accepts only IP packets by default. You must configure the router interfaces to recognize other protocol types. In this case, each interface must be aware that MPLS packets are important and should be accepted as well. To do this, you use the `family mpls` command, as in this example:

```
[edit]
user@Cabernet# set interfaces so-0/0/0 unit 0 family mpls
```

The `show interfaces terse` command reveals that Cabernet’s interface is now correctly configured:

```
lab@Cabernet> show interfaces terse
Interface       Admin Link Proto Local                Remote
so-0/0/0        up    up
so-0/0/0.0      up    up   inet  10.100.10.1/24
                           iso
                           mpls
so-0/0/1        up    up
so-0/0/2        up    up
so-0/0/3        up    down
fxp0            up    up
fxp0.0          up    up   inet  10.250.0.121/16
fxp1            up    up
fxp1.0          up    up   tnp   4
gre             up    up
ipip            up    up
lo0             up    up
lo0.0           up    up   inet  192.168.1.1         --> 0/0
                           iso   49.1111.0192.0168.0101.00
lsi             up    up
```

Configuring the Protocol

From the perspective of the Routing Engine, MPLS is just another protocol. As such, it requires some configuration within the `[edit protocols]` hierarchy. Within this directory, you must assign each transit interface that supports MPLS.

Let's suppose the Muscat router wants all transit interfaces to support MPLS traffic. The configuration looks like this:

```
[edit protocols]
user@Muscat# show
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

Once you commit the configuration, the router creates the `mpls.0` routing table and places three entries in it. You can see this by issuing the `show route table mpls.0` command:

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

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

0                  *[MPLS/0] 00:11:26, metric 1
                     Receive
1                  *[MPLS/0] 00:11:26, metric 1
                     Receive
2                  *[MPLS/0] 00:11:26, metric 1
                     Receive
```

Referring back to the "Labels" section earlier in this chapter, the three preestablished entries correspond to the IPv4 Explicit NULL (0), the Router Alert label (1), and the IPv6 Explicit NULL (2). Notice that each label has a next-hop value of `Receive`. This sends all packets matching this value to the Routing Engine for further processing.

This next hop makes sense if you recall the meaning of these special label values. The 0 label means that the receiving router should perform a label pop operation and then perform an IPv4 route lookup. The 1 label requires the receiving router to process the packet by the routing process and not perform a forwarding table lookup. The 2 label means that the receiving router should perform a label pop operation and then perform an IPv6 route lookup.

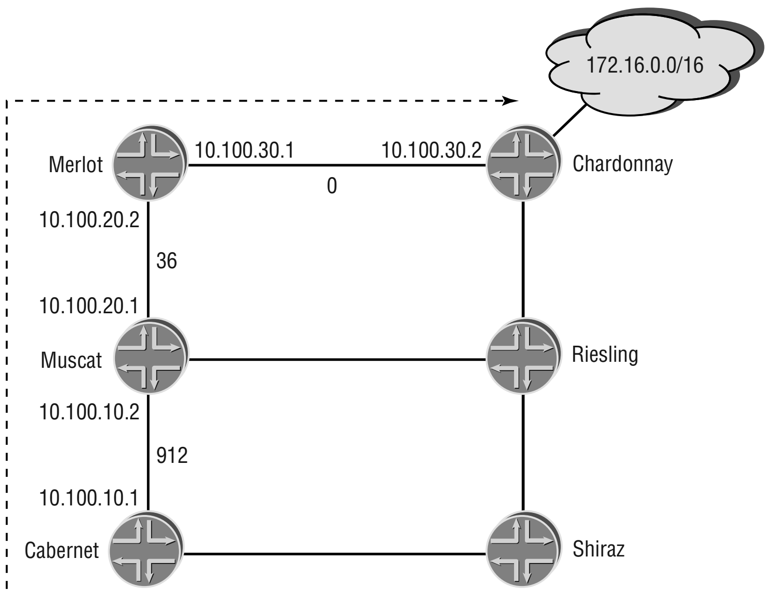
NOTE

In reality, when a Juniper Networks router receives a label value of 0, the packet does not go to the Routing Engine. All processing is instead completed in hardware. The incoming I/O Manager ASIC on the Flexible PIC Concentrator (FPC) performs the label pop operation. The resulting IP packet is then turned into J-Cells and stored in packet memory. The Internet Processor ASIC performs a regular IPv4 route lookup on the packet and forwards it to the appropriate next hop.

Configuring the Static LSP

We are now ready to configure the actual LSP. We would like the LSP to operate between Cabernet and Chardonnay, as shown in Figure 11.16. You can see the label values assigned to the links. Cabernet uses label 912 to forward packets to Muscat. Muscat uses label 36 to forward packets to Merlot. Merlot uses label 0 when sending packets to Chardonnay, the egress router.

FIGURE 11.16 Static LSP label assignments



NOTE

The IETF reserved label values 0 through 15. The JUNOS software sets aside label values 16 through 1023 for use with static LSPs. A Juniper Networks router can use a dynamic advertisement from a downstream neighbor from within this range. This range simply means that the JUNOS software will never use those values when generating its own dynamic label values.

Next, we configure the ingress router, followed by the two transit routers. The egress router does not need any special configuration at this point. It performs a label pop when it receives the 0 label from the penultimate router.

Configuring the Ingress Router

We want Cabernet to use MPLS to forward traffic to the 172.16.0.0 /16 subnet attached to Chardonnay. The following command accomplishes this goal:

```
[edit protocols mpls]
user@Cabernet# set static-path inet 172.16/16 push 912 next-hop 10.100.10.2
```

The configuration now looks like this:

```
[edit protocols mpls]
user@Cabernet# show
static-path inet {
    172.16.0.0/16 {
        next-hop 10.100.10.2;
        push 912;
    }
}
interface all;
```

The MPLS-specific information for this route now appears in the `inet.0` routing table on Cabernet, as shown here:

```
user@Cabernet> show route table inet.0

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

10.100.10.0/24     *[Direct/0] 00:42:17
                    > via so-0/0/0.0
10.100.10.1/32     *[Local/0] 00:45:57
                     Local via so-0/0/0.0
10.100.20.0/24     *[IS-IS/15] 00:34:41, metric 20, tag 1
                    > to 10.100.10.2 via so-0/0/0.0
10.100.30.0/24     *[IS-IS/15] 00:33:45, metric 30, tag 1
                    > to 10.100.10.2 via so-0/0/0.0
172.16.0.0/16      *[Static/5] 00:00:33
                    > to 10.100.10.2 via so-0/0/0.0, Push 912
192.168.1.1/32     *[Direct/0] 00:45:57
                    > via lo0.0
```

```
192.168.2.2/32     *[IS-IS/15] 00:41:50, metric 10, tag 1
                    > to 10.100.10.2 via so-0/0/0.0
192.168.3.3/32     *[IS-IS/15] 00:34:22, metric 20, tag 1
                    > to 10.100.10.2 via so-0/0/0.0
192.168.4.4/32     *[IS-IS/15] 00:33:22, metric 30, tag 1
                    > to 10.100.10.2 via so-0/0/0.0
```

All route lookups for 172.16.0.0 /16 result in label 912 being added to the packet. The resulting MPLS information is forwarded to Muscat on the interface `so-0/0/0.0`.

NOTE: A static LSP is assigned to the `Static` protocol within the routing table, making it no different from a regular static route.

Configuring the Transit Routers

The configuration on each of the transit routers is similar. The incoming interface and label value must be identified. Let's configure the resulting label operation first, followed by the next hop of the LSP.

We configure Muscat first. It should watch for label 912 on its interface to Cabernet—`so-0/0/0.0`. Muscat should swap label 912 with label 36 and send the packet to Merlot's interface of 10.100.20.2. We use the following commands to accomplish this:

```
[edit protocols mpls]
user@Muscat# set interface so-0/0/0 label-map 912 swap 36
user@Muscat# set interface so-0/0/0 label-map 912 next-hop 10.100.20.2
```

The resulting configuration now appears as:

```
[edit protocols mpls]
user@Muscat# show
interface all;
interface so-0/0/0.0 {
    label-map 912 {
        next-hop 10.100.20.2;
        swap 36;
    }
}
```

The Merlot router is similarly configured as:

```
[edit protocols mpls]
user@Merlot# show
interface all;
interface so-0/0/0.0 {
    label-map 36 {
        next-hop 10.100.30.2;
        swap 0;
    }
}
```

Once committed, the label operations are visible in the `mpls.0` table:

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

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

0                  *[MPLS/0] 00:49:20, metric 1
                      Receive
1                  *[MPLS/0] 00:49:20, metric 1
                      Receive
2                  *[MPLS/0] 00:49:20, metric 1
                      Receive
36                 *[Static/5] 00:00:06
                    > to 10.100.30.2 via so-0/0/2.0, Swap 0
```

NOTE The addresses used in the `next-hop` command represent the address of a directly connected router. This is an important concept since recursive IP route lookups are not performed inside the LSP.

Verifying the Operation

We've already seen that Cabernet has an IP route in `inet.0` for the 172.16.0.0 /16 subnet. However, like a static route, this does not prove end-to-end connectivity. Let's see if the ingress router can ping the subnet:

```
user@Cabernet> ping 172.16.1.1
PING 172.16.1.1 (172.16.1.1): 56 data bytes
64 bytes from 172.16.1.1: icmp_seq=0 ttl=252 time=1.059 ms
64 bytes from 172.16.1.1: icmp_seq=1 ttl=252 time=0.976 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=252 time=0.941 ms
^C
--- 172.16.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.941/0.992/1.059/0.049 ms
```

Things look good so far. It is interesting to see the information gathered with the `traceroute` command:

```
user@Cabernet> traceroute 172.16.1.1
traceroute to 172.16.1.1 (172.16.1.1), 30 hops max, 40 byte packets
 1  10.100.10.2 (10.100.10.2)  0.904 ms  0.700 ms  0.657 ms
     MPLS Label=912 CoS=0 TTL=1 S=1
 2  10.100.20.2 (10.100.20.2)  0.753 ms  0.706 ms  0.676 ms
     MPLS Label=36 CoS=0 TTL=1 S=1
 3  10.100.30.2 (10.100.30.2)  0.721 ms  0.675 ms  0.651 ms
 4  172.16.1.1 (172.16.1.1)  0.889 ms  0.750 ms  0.730 ms
```

Note that the MPLS label information is returned as part of the output. These ICMP messages are returned using IP routing lookups and truly prove that our data packets are using the LSP to reach the 172.16.1.1 address.

Please remember that static LSPs do not have a keepalive mechanism. As such, there is no way to view the LSP to verify its status. The ingress router could start forwarding packets to the LSP even if the other routers are not configured. This will cause packet loss on the transit routers. You must be careful to coordinate your efforts when configuring static LSPs.

Configuring a Dynamic LSP

One advantage of establishing a dynamic LSP is that only the ingress router requires configuration knowledge of the LSP; the other routers in the network do not need any explicit configuration. This is true for the LSP itself; however, all routers in the network do need information about MPLS and the signaling protocols in general. This allows for the setup of the LSP using RSVP `Path` and `Resv` messages.

Let's now look at the steps needed to support RSVP as a dynamic signaling protocol. Like the establishment of static LSPs, each router interface must use `family mpls` and the MPLS protocol needs to be enabled on the router. Each router in the network also enables the RSVP signaling protocol on all required interfaces. From there, the actual LSP is configured on the ingress router. First, we examine a basic configuration and then show you how to use bandwidth requests and assign an explicit route object.

Configuring Interfaces

The interfaces on all MPLS routers in the network should now accept more than just IP packets, which is the default. Each interface should accept and process MPLS packets using the `family mpls` command. The Merlot router accomplishes this goal:

```
[edit]
```

```
user@Merlot# set interfaces so-0/0/0 unit 0 family mpls
user@Merlot# set interfaces so-0/0/2 unit 0 family mpls
```

The show `interfaces terse` command demonstrates our successful configuration:

```
lab@Merlot> show interfaces terse
Interface       Admin Link Proto Local                 Remote
so-0/0/0        up    up
so-0/0/0.0      up    up   inet  10.100.20.2/24
                           iso
                           mpls
so-0/0/1        up    up
so-0/0/2        up    up
so-0/0/2.0      up    up   inet  10.100.30.1/24
                           iso
                           mpls
so-0/0/3        up    down
fxp0            up    up
fxp0.0          up    up   inet  10.250.0.123/16
fxp1            up    up
fxp1.0          up    up   tnp   4
gre             up    up
ipip            up    up
lo0             up    up
lo0.0           up    up   inet  192.168.3.3         --> 0/0
                           iso   49.1111.0192.0168.0303.00
lsi             up    up
```

Configuring the MPLS Protocol

Each transit interface that supports MPLS requires some configuration within the `[edit protocols]` hierarchy. In this example, we want all transit interfaces on the Chardonnay router to support MPLS traffic:

```
[edit protocols]
user@Chardonnay# show
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

As we saw in the “Configuring a Static LSP” section earlier in this chapter, the router creates the `mpls.0` routing table once MPLS is created as a protocol. We can see this table by using the `show route table mpls.0` command:

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

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

0                  *[MPLS/0] 14:53:33, metric 1
                    Receive
1                  *[MPLS/0] 14:53:33, metric 1
                    Receive
2                  *[MPLS/0] 14:53:33, metric 1
                    Receive
```

Configuring the RSVP Protocol

RSVP is another protocol enabled on the Routing Engine to support dynamic LSPs. Each interface receiving and sending RSVP messages requires a configuration in the `[edit protocols]` hierarchy. A typical router places the same interfaces in both the MPLS and RSVP configuration directories.

Chardonnay now adds its interfaces to the RSVP section of the configuration. The result of the change is:

```
[edit protocols]
user@Chardonnay# show
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

NOTE: Many people find the `all` keyword a bit confusing. Which interfaces does it find? How do you know it will be running properly? The JUNOS software does provide some `show` commands to answer those questions. However, from the perspective of MPLS and RSVP, only interfaces configured with `family mpls` are activated for those protocols. In our case, we do want all interfaces, except `fxp0.0`, using the protocols. In your network, you may want to explicitly list interfaces in the protocol configuration. Both solutions are acceptable.

We can check the current status of our interfaces to ensure that each router is supporting both MPLS and RSVP on all interfaces. The Muscat router has been completed, so we'll check it now:

```
user@Muscat> show mpls interface
Interface        State       Administrative groups
so-0/0/0.0       Up         <none>
so-0/0/2.0       Up         <none>

user@Muscat> show rsvp interface
RSVP interface: 2 active
                  Active Subscr- Static      Available   Reserved    Highwater
Interface    State resv   iption BW          BW          BW          mark
so-0/0/0.0   Up         0   100% 155.52Mbps  155.52Mbps  0bps        0bps
so-0/0/2.0   Up         0   100% 155.52Mbps  155.52Mbps  0bps        0bps
```

Both of the transit interfaces are reporting a state of `Up`. This is the desired state we're looking for. We discuss the bandwidth information in the `show rsvp interface` output in the "Configuring LSP Attributes" section later in this chapter.

Configuring the Dynamic LSP

We're now ready to configure the LSP on the ingress router. We would like the LSP to follow the current IGP shortest path between Cabernet and Chardonnay. The desired network path is shown in Figure 11.17.

Configuring the Ingress Router

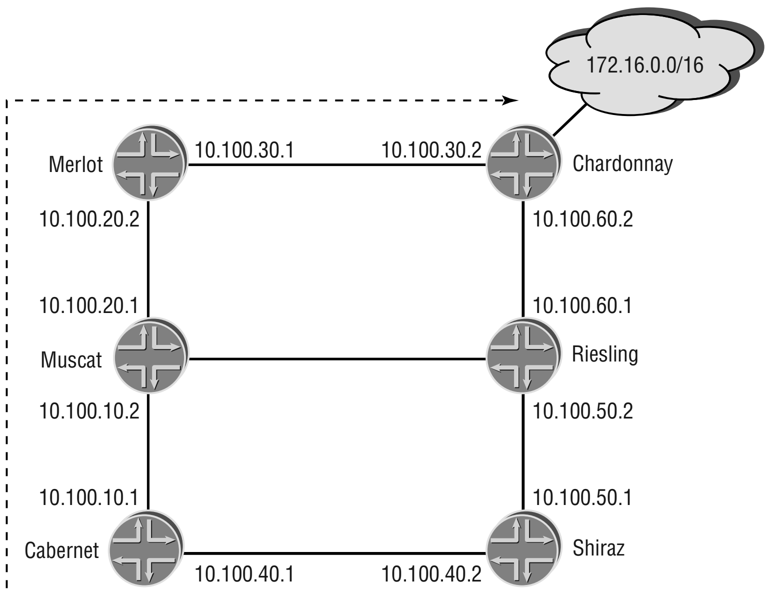
The minimum configuration of an *RSVP signaled LSP* requires an ASCII name and the address of the egress router. Suppose you issued the following commands on Cabernet:

```
[edit protocols mpls]
user@Cabernet# set label-switched-path Cab-to-Char to 192.168.4.4
user@Cabernet# set label-switched-path Cab-to-Char no-cspf
```

The configuration on Cabernet now looks like this:

```
[edit protocols mpls]
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    no-cspf;
}
interface all;
interface fxp0.0 {
    disable;
}
```

FIGURE 11.17 Dynamic LSP network path



NOTE

The JUNOS software default for dynamic LSPs is to calculate the path using a Traffic Engineering Database (TED) and the Constrained Shortest Path First (CSPF) algorithm. We have decided that the path should be calculated on a hop-by-hop basis using the IGP routing tables on each router. The `no-cspf` command disables the default use of the CSPF algorithm. The CSPF algorithm and the Traffic Engineering Database are discussed in the *JNCIS: Juniper Networks Certified Internet Specialist Study Guide.*
.

To verify the establishment of the LSP, we can use the `show mpls lsp` command on any router. We first check it on the ingress router, Cabernet:

```
user@Cabernet> show mpls lsp
Ingress LSP: 1 sessions
To             From           State Rt ActivePath       P     LSPname
192.168.4.4    192.168.1.1    Up     0                  *     Cab-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
```

It appears as if the ***Cab-to-Char*** LSP has an operational status of `Up`. Notice that the command output also displays LSPs for which the local router is an egress or transit router. A Juniper Networks router can perform all these functions simultaneously, so this output makes sense. By adding the *`extensive`* option to the command on the ingress router, we see some other useful information:

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

192.168.4.4
  From: 192.168.1.1, State: Up, ActiveRoute: 0, LSPname: Cab-to-Char
  ActivePath:  (primary)
  LoadBalance: Random
 *Primary                    State: Up
    Received RRO:
          10.100.10.2 10.100.20.2 10.100.30.2
    4 Jul 11 08:44:41  Selected as active path
    3 Jul 11 08:44:41  Record Route:  10.100.10.2 10.100.20.2 10.100.30.2
    2 Jul 11 08:44:41  Up
    1 Jul 11 08:44:41  Originate Call
   Created: Thu Jul 11 08:39:18 2002
Total 1 displayed, Up 1, Down 0

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

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

The record route object (RRO) from the `Resv` message is visible in this display. This shows us the exact path the LSP is taking through the network. A quick look at Figure 11.17 reveals that the LSP follows the desired path of Cabernet, Muscat, Merlot, and Chardonnay.

Once a dynamic LSP is established and usable, the router inserts information about that LSP in the routing table structure. Specifically, the router places the egress address of the LSP in the `inet.3` routing table. We can verify this with the `show route` command:

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

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.4.4/32     *[RSVP/7] 01:05:55, metric 30, metric2 0
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

The 192.168.4.4 /32 address is the loopback IP address of Chardonnay, so it appears the defaults are working as intended.

Configuring Transit and Egress Routers

No explicit configuration is required on the transit and egress routers to set up a dynamic LSP. After establishing the LSP, you can check its status on these routers by examining the RSVP soft state. Let's check one of the transit routers:

```
user@Muscat> 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
192.168.4.4     192.168.1.1     Up     1  1 FF  100000   100000 Cab-to-Char
Total 1 displayed, Up 1, Down 0
```

Again, the LSP is operational. This display shows the label values that Muscat received from Merlot (`Labelout`) and sent to Cabernet (`Labelin`) for this LSP. This is useful for troubleshooting an LSP's behavior.

NOTE

The label value of 100000 appearing on both the `Labelin` and `Labelout` fields might be a bit confusing. The JUNOS software allocates dynamic LSP labels in the 100,000 to 1,048,576 range. As you start your MPLS configuration, you will notice that many label values appear on multiple links for the same LSP. This is what happened here. As the number of LSPs and routers grows in your network, this "pattern" disappears and the label values become more randomized.

Finally, we verify the egress router:

```
user@Chardonnay> show mpls lsp
Ingress LSP: 0 sessions
Total 0 displayed, Up 0, Down 0

Egress LSP: 1 sessions
To              From            State Rt Style Labelin Labelout LSPname
192.168.4.4     192.168.1.1     Up     0  1 FF       3        - Cab-to-Char
Total 1 displayed, Up 1, Down 0

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

Notice that Chardonnay lists a label value of 3 in the `Labelin` field. Recall from the "Labels" section earlier in this chapter that a value of 3 represents an Implicit NULL. This is the default behavior for the JUNOS software and means that the penultimate router (Merlot) is performing penultimate hop popping (PHP). We can verify this behavior by examining the Merlot output:

```
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
192.168.4.4     192.168.1.1     Up     1  1 FF  100000        3 Cab-to-Char
Total 1 displayed, Up 1, Down 0
```

Merlot has correctly listed a label value of 3 in the `Labelout` field to support PHP.

Configuring LSP Attributes

At this point, we would like to control the path of the LSP through the network. To accomplish this, we configure an ERO and assign it to the LSP. We also add an RSVP bandwidth request of 15Mbps to the LSP. Since a dynamic LSP is configured only on the ingress router, we make these changes on Cabernet.

An ERO is called a *named path* in the JUNOS software. This is where loose and strict hops are defined. Using Figure 11.17 as a guide, the LSP should now traverse along the Cabernet, Shiraz, Riesling, and Chardonnay routers. We first build the path and define Shiraz as a loose hop:

```
[edit protocols mpls]
user@Cabernet# set path via-Shiraz 192.168.5.5 loose
```

The configuration now looks like this:

```
[edit protocols mpls]
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    no-cspf;
}
path via-Shiraz {
    192.168.5.5 loose;
}
interface all;
```

The ERO called ***via-Shiraz*** tells the routers in the network to use the IGP routing tables to forward the `Path` message from the ingress router to the address of 192.168.5.5 (the loopback address of the Shiraz router). From there, the routers again consult their IGP routing tables to send the `Path` message from Shiraz to the egress router.

At this point, we use the *primary* option to assign the path to the LSP itself to ensure that the path takes effect. In short, we are telling the ingress router to use this named path as its preferred method for sending `Path` messages:

```
[edit protocols mpls]
user@Cabernet# set label-switched-path Cab-to-Char primary via-Shiraz
```

This alters the configuration:

```
[edit protocols mpls]
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    no-cspf;
    primary via-Shiraz;
}
path via-Shiraz {
    192.168.5.5 loose;
}
interface all;
```

The bandwidth request is now added to the LSP configuration:

```
[edit protocols mpls]
user@Cabernet# set label-switched-path Cab-to-Char bandwidth 15m
 [edit protocols mpls]
user@Cabernet# show
```

```
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    bandwidth 15m;
    no-cspf;
    primary via-Shiraz;
}
path via-Shiraz {
    192.168.5.5 loose;
}
interface all;
```

NOTE The JUNOS software uses bits/second as its default value for an LSP's bandwidth. You may alter this default by using the `k`, `m`, and `g` characters to represent kilobits, megabits, and gigabits per second, respectively. In our example, the router translates `15m` into 15Mbps of bandwidth.

After committing the configuration, we verify the operational status of the LSP by using the `show mpls lsp` command:

```
user@Cabernet> show mpls lsp
Ingress LSP: 1 sessions
To              From            State Rt ActivePath       P     LSPname
192.168.4.4     192.168.1.1     Up     0 via-Shiraz       *     Cab-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
```

It appears that our configuration was successful. The LSP is in an `Up` state and the current `ActivePath` is `via-Shiraz`. We can use the *`extensive`* option to see if the desired network path was taken:

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

192.168.4.4
  From: 192.168.1.1, State: Up, ActiveRoute: 0, LSPname: Cab-to-Char
  ActivePath: via-Shiraz (primary)
  LoadBalance: Random
```

```
 *Primary   via-Shiraz       State: Up
    Bandwidth: 15Mbps
    Received RRO:
          10.100.40.2 10.100.50.2 10.100.60.2
    4 Jul 11 11:39:44  Selected as active path
    3 Jul 11 11:39:44  Record Route:  10.100.40.2 10.100.50.2 10.100.60.2
    2 Jul 11 11:39:44  Up
    1 Jul 11 11:39:44  Originate Call
   Created: Thu Jul 11 11:37:27 2002
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
```

Based on the `Resv` message RRO, our desired path was selected. This output also displays the bandwidth request of 15Mbps for the LSP. We can verify that the network honored this request by examining a transit router. The `show rsvp interface` command details the current reservations in the network:

```
user@Shiraz> show rsvp interface
RSVP interface: 2 active
                  Active Subscr- Static      Available   Reserved    Highwater
Interface    State resv   iption BW          BW          BW          mark
so-0/0/0.0   Up        0    100% 155.52Mbps  155.52Mbps  0bps        0bps
so-0/0/2.0   Up        1    100% 155.52Mbps  140.52Mbps  15Mbps      15Mbps
```

The `so-0/0/2.0` interface shows a current `Reserved BW` of 15Mbps. This is the interface headed to the Riesling router and in the direction of the egress. Remember that LSPs are unidirectional in nature and, therefore, reservations flow in a downstream direction.

Altering the RSVP Protocol

The JUNOS software provides several configuration options you can use to alter the behavior of the RSVP protocol. We quickly talk about what each knob does and look at a configuration example for each.

Hello Interval Timer

The JUNOS software uses the extended RSVP option of sending hello messages to its neighbors. The hello mechanism speeds the detection of router failures in the network. Without it, each router waits for the RSVP soft state to expire before detecting an outage.

For backward compatibility, hello messages are asynchronous in nature. Each router uses the timer advertised from its peer to calculate the hold interval for that neighbor. Should a peer not send hello messages, the local router relies on the soft state for expiring the session. This provides backward compatibility with the original RSVP specification.

The default interval for hello messages is 9 seconds, with a possible range of 1 to 60 seconds. Each router determines the hold interval for each neighbor. When (2 × `keep-multiplier` + 1) consecutive hello messages are not received, the neighbor is declared dead. By default, the `keep-multiplier` value is 3, which leads to 7 missed hello messages. Using the hello interval of 9 seconds results in a dead interval of 63 seconds for each neighbor.

NOTE

RSVP neighbor loss is also detected by a physical layer interface change. When the interface goes down, the RSVP state is removed immediately. The 63-second hold time is used when the RSVP process itself stops operating on the neighboring router.

We can see the hello interval in the output of the `show rsvp neighbor` command. The Chardonnay router is using the default values:

```
user@Chardonnay> show rsvp neighbor
RSVP neighbor: 2 learned
Address           Idle Up/Dn LastChange HelloInt HelloTx/Rx MsgRcvd Status
10.100.30.1          0  1/0     5:04:40        9  5940/5940     238 -
10.100.60.1          0  1/0     2:09:37        9  2529/2529     177 -
```

The interval is changed on Chardonnay's `so-0/0/2.0` interface (10.100.60.1) to 20 seconds:

```
[edit protocols rsvp]
user@Chardonnay# set interface so-0/0/2 hello-interval 20

[edit protocols rsvp]
user@Chardonnay# show
interface all;
interface fxp0.0 {
    disable;
}
interface so-0/0/2.0 {
    hello-interval 20;
}

user@Chardonnay> show rsvp neighbor
RSVP neighbor: 2 learned
Address           Idle Up/Dn LastChange HelloInt HelloTx/Rx MsgRcvd Status
10.100.30.1          5  1/0     5:09:06        9  6027/6027     238 -
10.100.60.1          5  1/0     2:14:03       20  2616/2616     183 -
```

Soft State Refresh Timer

Without the use of the hello mechanism, RSVP routers use the soft state timers to detect network outages. Each router sends `Path` and `Resv` messages to its neighbors for each current LSP known to the local router. These messages are sent only between adjacent nodes and do not travel the length of the LSP.

The JUNOS software uses a default value of 30 seconds, with a possible range between 1 and 65,535 seconds. The refresh value is used to calculate the lifetime of the RSVP soft state database. The total lifetime is found using the formula (`keep-multiplier` + 0.5) × 1.5 × `refresh-time`. The default values mean that each router waits 157.5 seconds (2.625 minutes), as a worst case, before declaring a network outage.

NOTE The actual time for sending an RSVP refresh message ranges from 15 seconds (0.5 × `refresh-time`) to 45 seconds (1.5 × `refresh-time`).

Chardonnay reduces its `refresh-time` to 15 seconds to speed the expiration of the soft state. This option is configurable at the global RSVP level only.

```
[edit protocols rsvp]
user@Chardonnay# set refresh-time 15

[edit protocols rsvp]
user@Chardonnay# show
refresh-time 15;
interface all;
interface fxp0.0 {
    disable;
}
interface so-0/0/2.0 {
    hello-interval 20;
}
```

We can see the result of this configuration in the output of `show rsvp version`:

```
user@Chardonnay> show rsvp version
Resource ReSerVation Protocol, version 1. rfc2205
   RSVP protocol      = Enabled
   R(refresh timer)   = 15 seconds
   K(keep multiplier) = 3
   Preemption         = Normal
```

Multiplier Value

You can use the `keep-multiplier` command to calculate timer values for network outages. Both the hello and soft state timers use the multiplier to determine when to notify neighbors about a failure. The JUNOS software default value for the multiplier is 3, with a possible range between 1 and 255.

To speed detection of outages, Chardonnay alters the `keep-multiplier` for its RSVP process:

```
[edit protocols rsvp]
user@Chardonnay# set keep-multiplier 1

[edit protocols rsvp]
user@Chardonnay# show
refresh-time 15;
keep-multiplier 1;
interface all;
interface fxp0.0 {
    disable;
}
interface so-0/0/2.0 {
    hello-interval 20;
}
```

The `show rsvp version` command shows the effect of this configuration:

```
user@Chardonnay> show rsvp version
Resource ReSerVation Protocol, version 1. rfc2205
   RSVP protocol      = Enabled
   R(refresh timer)   = 15 seconds
   K(keep multiplier) = 1
   Preemption         = Normal
```

Message Aggregation

Each RSVP message sent between neighbors is contained in a separate packet. This includes hello, `Path`, `Resv`, and error messages. Your router can negotiate with its neighbors on an interface-by-interface basis to bundle up to 30 RSVP messages (non-configurable) in a single packet before sending it to that neighbor. This greatly reduces the overhead of running RSVP and provides for greater network scalability.

As you can see here, Chardonnay enables this feature on all of its transit interfaces:

```
[edit protocols rsvp]
user@Chardonnay# set interface so-0/0/2.0 aggregate
```

```
[edit protocols rsvp]
user@Chardonnay# set interface all aggregate

[edit protocols rsvp]
user@Chardonnay# show
refresh-time 15;
keep-multiplier 1;
interface all {
    aggregate;
}
interface fxp0.0 {
    disable;
}
interface so-0/0/2.0 {
    aggregate;
    hello-interval 20;
}
```

TIP

Notice that the `aggregate` command is configured twice on the Chardonnay router. Keep in mind that the JUNOS software uses the most specific reference to a command possible. Since the `so-0/0/2.0` interface is listed separately, it doesn't inherit commands applied within the `interface all` portion of the configuration. This requires us to use the two commands shown to apply the `aggregate` option to all the transit interfaces.

Authenticating RSVP

You can authenticate RSVP message exchanges between routers. To do so, you use the MD5 authentication mechanism with a key length of 16 characters.

In our example, Chardonnay configures authentication on its `so-0/0/2.0` interface with a key of ***password***:

```
[edit protocols rsvp]
user@Chardonnay# set interface so-0/0/2 authentication-key password

[edit protocols rsvp]
user@Chardonnay# show
refresh-time 15;
keep-multiplier 1;
interface all {
    aggregate;
}
```

```
interface fxp0.0 {
    disable;
}
interface so-0/0/2.0 {
    authentication-key "$9$2SgZjHkPQ39.PhrvLVb.P5Tz6"; # SECRET-DATA
    aggregate;
    hello-interval 20;
}
```

Bandwidth Reservation Limits

The JUNOS software allocates 100 percent of the physical interface bandwidth for RSVP reservations. You have two methods for altering this default behavior. The first is changing the reservation percentage for the entire physical interface. The second is specifying an actual bandwidth value to be used for a logical interface unit. This second method is very useful for controlling reservations on ATM and Frame Relay circuits.

Chardonnay is currently using the default reservation percentage. We can see this by issuing the `show rsvp interface` command:

```
user@Chardonnay> show rsvp interface
RSVP interface: 2 active
                  Active Subscr- Static      Available   Reserved    Highwater
Interface   State resv    iption BW          BW          BW          mark
so-0/0/0.0  Up         0   100%  155.52Mbps  155.52Mbps  0bps        0bps
so-0/0/2.0  Up         0   100%  155.52Mbps  155.52Mbps  0bps        0bps
```

Both methods are used to alter the default reservation availability:

```
[edit protocols rsvp]
user@Chardonnay# set interface so-0/0/0 bandwidth 20m

[edit protocols rsvp]
user@Chardonnay# set interface so-0/0/2 subscription 65

[edit protocols rsvp]
user@Chardonnay# show
refresh-time 15;
keep-multiplier 1;
interface all {
    aggregate;
}
interface fxp0.0 {
    disable;
}
```

```
interface so-0/0/2.0 {
    authentication-key "$9$2SgZjHkPQ39.PhrvLVb.P5Tz6"; # SECRET-DATA
    subscription 65;
    aggregate;
    hello-interval 20;
}
interface so-0/0/0.0 {
    bandwidth 20m;
}
```

We then verify the results:

```
user@Chardonnay> show rsvp interface
RSVP interface: 2 active
                  Active Subscr- Static      Available   Reserved    Highwater
Interface   State resv   iption  BW          BW          BW          mark
so-0/0/0.0  Up         0   100%  20Mbps      20Mbps      0bps        0bps
so-0/0/2.0  Up         0    65%  155.52Mbps  101.088Mbps 0bps        0bps
```

The values in the `Subscription` and `Static BW` columns for non-configured options remain unchanged. The result of the configuration is best seen by adding the `Available BW` and `Reserved BW` columns together. As there are no current LSP bandwidth reservations on Chardonnay, we see that the `so-0/0/0.0` interface is using the static value of 20Mbps, and the `so-0/0/2.0` interface calculated 65 percent of the physical bandwidth for a result of 101.088Mbps.

Routing Table Integration

In the "Configuring a Static LSP" section earlier in this chapter, we explicitly assigned the 172.16.0.0 /16 subnet to use the LSP across the network. We made no such routing association in the "Configuring a Dynamic LSP" section earlier in this chapter. This was not an oversight on our part. The JUNOS software has a default action for associating IP routes to established dynamic LSPs. In addition, numerous configuration options are available for changing the default behavior. As such, this topic receives its own special treatment at this point.

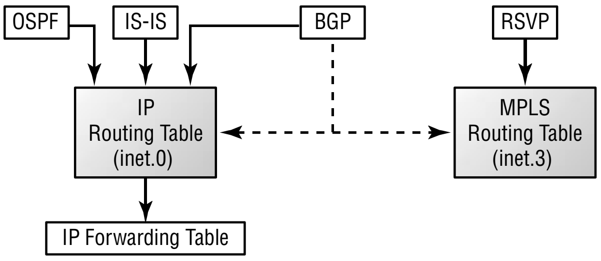
We first explore how a Juniper Networks router uses LSPs by default. We then describe a few methods for altering the default behavior.

Default Behavior

There is an inherent relationship between LSPs and BGP. The JUNOS software assumes that traffic using BGP routes should also use the LSP for data forwarding. This is useful for engineering transit user traffic across your network.

The BGP route and the established LSP are linked together when the route is installed in the routing table. Recall from Chapter 8, "Border Gateway Protocol," that a BGP route can be used only when the BGP Next Hop attribute is reachable. Under normal circumstances, the BGP process examines the `inet.0` routing table to verify this reachability. When an LSP has been established and placed in the `inet.3` routing table, the BGP process can examine its contents to determine reachability to the BGP Next Hop. Figure 11.18 shows this process.

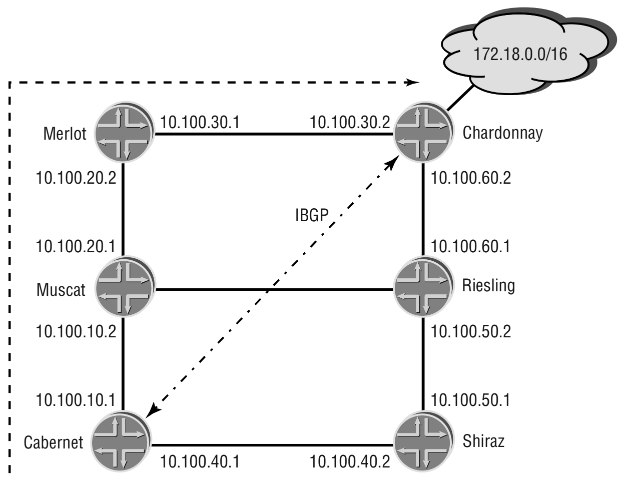
FIGURE 11.18 Default routing table integration



The RSVP information is installed in `inet.3`. The IGP routes from OSPF and IS-IS are installed in `inet.0`. As BGP attempts to place a route in `inet.0`, it examines both `inet.0` and `inet.3` during its BGP Next Hop resolution. This longest-match lookup process identifies the version of the route with the lowest JUNOS software preference. RSVP installs information with a preference value of 7, OSPF uses a value of 10, and IS-IS uses both 15 and 18. By default, then, the BGP route uses the established LSP as its physical forwarding path across the network. Let's examine this behavior in a sample network, shown in Figure 11.19.

NOTE In the case of a preference tie between the routing tables, the JUNOS software prefers the `inet.3` table and the LSP.

FIGURE 11.19 Route table integration network



The Cabernet router is peering with Chardonnay using IBGP routing. The 172.18.0.0 /16 route is advertised to Cabernet over this connection. The current BGP next-hop value is 192.168.4.4, the loopback address of Chardonnay. We can verify this information by using the `show route receive-protocol` command on Cabernet:

```
user@Cabernet> show route receive-protocol bgp 192.168.4.4

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

172.18.0.0/16
192.168.4.4             0       100 2 I
```

The next hop of 192.168.4.4 is currently assigned to the 172.18.0.0 /16 BGP route. No LSP has been established and Cabernet has reachability only to 192.168.4.4 through IS-IS:

```
user@Cabernet> show route 192.168.4.4

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

192.168.4.4/32     *[IS-IS/15] 07:00:15, metric 30, tag 1
                      to 10.100.10.2 via so-0/0/0.0
                    > to 10.100.40.2 via so-0/0/2.0
```

The BGP route is installed in the `inet.0` routing table with a next-hop value of 10.100.40.2 out the `so-0/0/2.0` interface:

```
user@Cabernet> show route 172.18/16

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

172.18.0.0/16      *[BGP/170] 00:04:34, MED 0, localpref 100, from 192.168.4.4
                      AS path: 2 I
                      to 10.100.10.2 via so-0/0/0.0
                    > to 10.100.40.2 via so-0/0/2.0
```

We have configured an LSP on Cabernet that uses Chardonnay as the egress router. The desired path through Muscat and Merlot in Figure 11.19 is enforced with a named path ERO. The egress router address now appears in the `inet.3` routing table:

```
[edit protocols mpls]
```

```
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    no-cspf;
    primary via-Merlot;
}
path via-Merlot {
    192.168.3.3 loose;
}
interface all;
interface fxp0.0 {
    disable;
}

user@Cabernet> show route table inet.3

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.4.4/32     *[RSVP/7] 00:00:03, metric 30, metric2 0
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

Cabernet now has reachability to 192.168.4.4 through both RSVP and IS-IS:

```
user@Cabernet> show route 192.168.4.4

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

192.168.4.4/32     *[IS-IS/15] 07:07:55, metric 30, tag 1
                      to 10.100.10.2 via so-0/0/0.0
                    > to 10.100.40.2 via so-0/0/2.0

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.4.4/32     *[RSVP/7] 00:01:30, metric 30, metric2 0
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

The BGP Next Hop resolution process finds both versions of the next hop and chooses the LSP because of a lower route preference. The 172.18.0.0 /16 is installed in the `inet.0` routing table with a next hop pointing to the LSP ***Cab-to-Char***.

```
user@Cabernet> show route 172.18/16

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

172.18.0.0/16      *[BGP/170] 00:12:22, MED 0, localpref 100, from 192.168.4.4
                      AS path: 2 I
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

NOTE: Should the LSP become unusable, the BGP Next Hop resolution process installs the route in `inet.0` with the IS-IS physical next hop, as before.

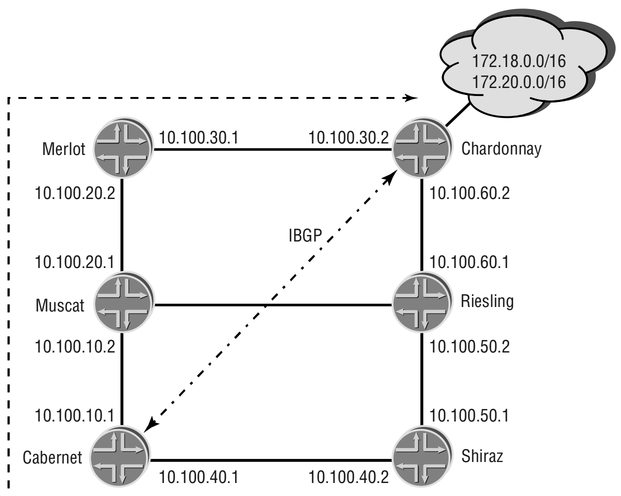
Assigning Individual Prefixes to an LSP

In addition to assigning BGP routes to an LSP automatically, you have the option of manually assigning routes. Perhaps the BGP next-hop address does not match the egress address of the LSP. Maybe you prefer to use the LSP to reach some internal non-BGP destination. Both of these scenarios are possible through the use of the `install` command.

Adding a Route to *inet.3*

The Chardonnay router has added the 172.20.0.0 /16 route to its routing table in Figure 11.20. Chardonnay then advertises the two BGP routes to Cabernet.

FIGURE 11.20 Assigning a prefix to an LSP



The 172.18.0.0 /16 route has a next hop of 192.168.4.4. The 172.20.0.0 /16 route has a next hop of 172.16.1.1:

```
user@Cabernet> show route receive-protocol bgp 192.168.4.4 all

inet.0: 18 destinations, 18 routes (17 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.18.0.0/16
192.168.4.4             0        100 2 I
172.20.0.0/16
172.16.1.1              0        100 2 I
```

Only the 172.18.0.0 /16 route is active in the `inet.0` routing table. Since Cabernet does not have reachability to 172.16.1.1, the 172.20.0.0 /16 route is currently hidden:

```
user@Cabernet> show route protocol bgp

inet.0: 18 destinations, 18 routes (17 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.18.0.0/16      *[BGP/170] 00:09:54, MED 0, localpref 100, from 192.168.4.4
                      AS path: 2 I
                    > via so-0/0/0.0, label-switched-path Cab-to-Char

user@Cabernet> show route 172.16.1.1

user@Cabernet> show route hidden

inet.0: 18 destinations, 18 routes (17 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

172.20.0.0/16       [BGP/170] 00:12:59, MED 0, localpref 100, from 192.168.4.4
                      AS path: 2 I
                      Unusable
```

The 172.16.1.1 address is assigned to the ***Cab-to-Char*** LSP. The `install` command places the address within the LSP configuration:

```
[edit protocols mpls]
user@Cabernet# set label-switched-path Cab-to-Char install 172.16.1.1
```

The configuration on Cabernet now looks like this:

```
[edit protocols mpls]
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    install 172.16.1.1/32;
    no-cspf;
    primary via-Merlot;
}
path via-Merlot {
    192.168.3.3 loose;
}
interface all;
interface fxp0.0 {
    disable;
}
```

The router places the 172.16.1.1 /32 address in the `inet.3` routing table as reachable through the ***Cab-to-Char*** LSP:

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

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

172.16.1.1/32      *[RSVP/7] 00:01:54, metric 30, metric2 0
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
192.168.4.4/32     *[RSVP/7] 00:01:54, metric 30, metric2 0
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

Cabernet has reachability to the BGP Next Hop of the 172.20.0.0 /16 BGP route and installs the route in the `inet.0` table:

```
user@Cabernet> show route 172.20/16

inet.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.20.0.0/16      *[BGP/170] 00:20:56, MED 0, localpref 100, from 192.168.4.4
                      AS path: 2 I
                    > via so-0/0/0.0, label-switched-path Cab-to-Char
```

Adding a Route to *inet.0*

You have decided that traffic destined for the 10.100.60.0 /24 route should use the LSP for data forwarding. This link is the Riesling-Chardonnay connection. Currently only an IS-IS route appears in the routing table:

```
user@Cabernet> show route 10.100.60/24

inet.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.100.60.0/24     *[IS-IS/15] 07:51:16, metric 30, tag 1
                    > to 10.100.40.2 via so-0/0/2.0
```

The next hop for the route is 10.100.40.2, Shiraz. In examining Figure 11.20, you find that the ***Cab-to-Char*** LSP currently follows the Cabernet, Muscat, Merlot, and Chardonnay path through the network. We assign the 10.100.60.0 /24 route to the ***Cab-to-Char*** LSP by using the `install` command:

```
[edit protocols mpls]
user@Cabernet# set label-switched-path Cab-to-Char install 10.100.60/24 active
```

The configuration now looks like this:

```
[edit protocols mpls]
user@Cabernet# show
label-switched-path Cab-to-Char {
    to 192.168.4.4;
    install 172.16.1.1/32;
    install 10.100.60.0/24 active;
    no-cspf;
    primary via-Merlot;
}
path via-Merlot {
    192.168.3.3 loose;
}
interface all;
interface fxp0.0 {
    disable;
}
```

We use the *`active`* option to install the specified prefix in the `inet.0` routing table instead of the `inet.3` routing table. This is a critical step in the process of allowing non-BGP routes to use the LSP. All user data traffic uses the `inet.0` table for route lookups. BGP routes in `inet.0` have a next hop of the LSP when RSVP installs the egress address in the `inet.3` table. Without the *`active`* keyword, only the `inet.3` table is updated and data traffic to the 10.100.60.0 /24 route still uses the IGP shortest path to Shiraz. We've avoided this problem and achieve our desired result.

```
user@Cabernet> show route 10.100.60/24

inet.0: 18 destinations, 19 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.100.60.0/24      *[RSVP/7] 00:05:20, metric 30, metric2 0
                     > via so-0/0/0.0, label-switched-path Cab-to-Char
                     [IS-IS/15] 08:01:02, metric 30, tag 1
                     > to 10.100.40.2 via so-0/0/2.0
```

User data traffic now uses the LSP to Muscat on the `so-0/0/0.0` interface.

Summary

In this chapter, we examined the basics of Multiprotocol Label Switching (MPLS). We started with a look at why the protocol was created and how its usefulness has evolved. Our discussion focused on the needs of ISPs to support growing bandwidth demands and the data-forwarding speed needed in that environment. We concluded our look at the history of the protocol by exploring the use of MPLS as a traffic-engineering mechanism.

We then described the terminology and conceptual aspect of MPLS. We defined an MPLS label and a label switched path (LSP), and discussed the functions of ingress, transit, penultimate, and egress routers.

We next examined the methods for establishing an LSP in an MPLS network. We looked at both static and dynamic methods. In addition, we explained how the Resource Reservation Protocol (RSVP) and Label Distribution Protocol (LDP) signaling protocols are used for the dynamic setup of an LSP.

We concluded the chapter by discussing the JUNOS software implementation of MPLS. We saw configuration examples of both static and RSVP signaled LSPs, described methods for altering the RSVP protocol, and discussed ways to place user data traffic into an LSP.

Exam Essentials

Understand the use of MPLS in an ISP network. MPLS is a method for engineering traffic flows across a network. It uses a switching table lookup to forward traffic based on a label value.

Be able to describe the functions of MPLS routers. Ingress routers add an MPLS label with a push operation. Transit routers swap label values and forward traffic along the LSP. The penultimate router performs a pop operation by removing the MPLS label and rewriting the TTL field back to the IP packet. The egress router terminates the LSP and forwards traffic based on an IP route lookup.

Identify the methods of LSP establishment. LSPs are established in one of two ways. Static LSPs are similar to static routes and require configuration of all routers. Dynamic LSPs use a signaling protocol to set up the LSP and require configuration on the ingress router only.

Be able to describe the signaling protocols used in MPLS. Both RSVP and LDP are valid protocols used to signal an LSP setup. The original RSVP specification was extended to support MPLS requirements. The LDP specification was created specifically for use with MPLS.

List the RSVP objects used for MPLS. The extended RSVP specification defines the explicit route object, label request object, label object, record route object, session attribute object, and tspec object.

Identify the steps required to configure a RSVP signaled LSP. The ingress router needs an ASCII name for the LSP and the address of the egress router. You may add other attributes, such as a bandwidth request and an explicit route object, to control the setup parameters of the LSP.

Be able to describe the integration of the JUNOS software routing tables. RSVP signaled LSPs place the egress router address in the `inet.3` routing table. BGP can use both the `inet.0` and `inet.3` tables to resolve reachability to the BGP Next Hop attribute. The longest-match route with the lowest preference value is used, which is the LSP by default.

Key Terms

Before you take the exam, be certain you are familiar with the following terms:

cell tax
dynamic label switched path
egress router
explicit route object (ERO)
hello mechanism
ingress router
Label Distribution Protocol (LDP)
label object
label pop operation
label push operation
label request object
label swap operation
label switched path (LSP)
label switching router (LSR)
label values
local significance
loose hops
message aggregation
named path
overlay network
`Path` message
`PathErr` message
`PathTear` message
penultimate hop popping (PHP)
penultimate router
record route object (RRO)
Resource Reservation Protocol (RSVP)
`ResvConf` message
`ResvErr` message
`ResvTear` message
RSVP `Resv` message
RSVP signaled LSP
session attribute object
shim header
soft state
static label switched path
strict hop
traffic engineering
transit routers
tspec object

Review Questions

1. What is MPLS used for in today's ISP networks?
   - **A.** ATM compatibility
   - **B.** Traffic engineering
   - **C.** Virtual Private Networks
   - **D.** Class of Service
2. What was the original goal of MPLS?
   - **A.** To increase the router's packet processing speed
   - **B.** To provide traffic engineering
   - **C.** To provide Layer 2 switching within Layer 3 routers
   - **D.** To provide Class of Service to IP traffic
3. What does LSR stand for?
   - **A.** Label Switching Router
   - **B.** Label Swapping Router
   - **C.** Layer 3 Switching Router
   - **D.** Layer 2 Swapping Router
4. What is the functionality of a transit router?
   - **A.** Performs MPLS encapsulation
   - **B.** Forwards traffic based on IP address
   - **C.** Performs a label swap operation
   - **D.** Forwards traffic based on MAC address
5. Where is an MPLS label placed in a packet on a Juniper Networks router?
   - **A.** Before the Layer 2 header
   - **B.** After the Layer 3 header
   - **C.** Between the Layers 2 and 3 headers
   - **D.** Between the Layers 3 and 4 headers
6. What MPLS router performs only a label push operation?
   - **A.** Ingress router
   - **B.** Transit router
   - **C.** Penultimate router
   - **D.** Egress router

7. What MPLS routers perform label swap operations? (Choose two.)

A. Ingress

B. Transit

C. Penultimate

D. Egress

8. How many bits form an MPLS header?

A. 16

B. 20

C. 22

D. 32

9. An MPLS header has a stack bit value of 0. What does this mean?

A. This is the last label.

B. There is only one more label in the stack.

C. The TTL has expired and the router needs to pop the label.

D. There are other labels in the stack.

10. What value is placed in the TTL field by the ingress LSR?

A. A value of 0.

B. A value of 255.

C. The IP TTL value is copied to the MPLS header.

D. The ingress LSR does nothing.

11. What does a label value of 2 mean?

A. IPv4 Explicit NULL

B. Router Alert

C. IPv6 Explicit NULL

D. Implicit NULL

12. Which protocols can be used for a dynamic LSP setup? (Choose two.)

A. IGP

B. LDP

C. BGP

D. RSVP

13. What label values are reserved by the IETF?

A. Labels 0 to 15

B. Labels 4 to 15

C. Labels 0 to 32

D. Labels 4 to 32

14. What router(s) require LSP configuration for a dynamic establishment?

A. Ingress

B. Penultimate

C. Transit

D. Egress

15. In which direction is a `Path` message sent?

A. Upstream to the ingress router

B. Downstream to the ingress router

C. Upstream to the egress router

D. Downstream to the egress router

16. In which direction is a `Resv` message sent?

A. Upstream to the ingress router

B. Downstream to the ingress router

C. Upstream to the egress router

D. Downstream to the egress router

17. How does RSVP maintain knowledge of the soft state of an LSP?

A. It uses RSVP authentication.

B. It uses RSVP hellos.

C. It refreshes RSVP `Path` and `Resv` messages.

D. It refreshes `PathErr` and `ResvErr` messages.

18. Into which JUNOS software routing table are RSVP signaled LSPs placed?

A. `inet.0`

B. `inet.1`

C. `inet.2`

D. `inet.3`

19. What types of hops are allowed in an ERO?

A. Strict hops only.

B. Loose hops only.

C. Both strict and loose hops.

D. No hops are allowed in an ERO.

20. How is the RSVP reservation information influenced when you configure `subscription` ***`percentage`*** on an interface?

A. It applies to the physical interface.

B. It applies to the entire router.

C. It applies to specific LSPs only.

D. It has no effect.

Answers to Review Questions

1. B. Traffic engineering is the main application of MPLS in today's networks.
2. A. The original IETF goal for MPLS was to increase the router's packet-processing speed.
3. A. LSR stands for Label Switching Router.
4. C. A transit MPLS router performs a label swap operation.
5. C. The JUNOS software uses a shim header that is placed between the Layers 2 and 3 headers.
6. A. The ingress router performs the label push operation in an LSP.
7. B, C. Transit and penultimate routers perform a label swap. The ingress and egress routers never perform this operation.
8. D. MPLS uses a 32-bit header.
9. D. When the S bit position contains a 0, it means that other labels exist in the stack.
10. C. By default, the ingress LSR copies the TTL value from the IP packet to the MPLS packet. The router that pops the MPLS label then copies the MPLS TTL back to the IP packet.
11. C. An MPLS label value of 2 represents an IPv6 Explicit NULL.
12. B, D. Both LDP and RSVP are signaling protocols supported by the JUNOS software.
13. A. The IETF reserved labels 0 to 15.
14. A. Only the ingress router needs specific LSP configuration in a dynamic environment.
15. D. `Path` messages are always sent in a downstream direction to the egress router.
16. A. `Resv` messages are always sent in an upstream direction to the ingress router.
17. C. RSVP refreshes `Path` and `Resv` messages every 30 seconds to maintain the soft state of the LSP in the network.
18. D. The egress address of RSVP LSPs are always placed in the `inet.3` routing table.
19. C. An RSVP ERO may contain both strict and loose hops.
20. A. The `subscription` ***percentage*** command applies to the entire physical interface.