CHAPTER

MPLS and IP Switching 17

What You Will Learn

In this chapter, you will learn how the desire for convergence has led to the development of various IP switching techniques. We'll also compare and contrast frame relay and ATM switched networks to illustrate the concepts behind IP switching.

You will learn how MPLS is used to create LSPs to switch (instead of route) IP packet through a routing domain. We'll see how MPLS can form the basis for a type of VPN service offering.

One of the reasons TCP/IP and the Internet have grown so popular is that this architecture is the promising way to create a type of "universal network" well suited for and equally at home with voice, video, and data. The Internet started as a network exclusively for data delivery, but has proved to be remarkably adaptable for different classes of traffic. Some say that more than half of all telephone calls are currently carried for part of their journey over the Internet, and this percentage will only go higher in the future. Why not watch an entire movie or TV show over the Internet? Many now watch episodes they missed on the Internet. Why not everything? As pointed out in the previous chapter, multicast might not be used to maximum effect for this but video delivery still works.

When a service provider adds television (or video in general) to Internet access and telephony, this is called a "triple play" opportunity for the service provider. (Adding wireless services over the Internet is sometimes called a "quadruple play" or "home run.")

This desire for networking convergence is not new. When the telephone was invented, there were more than 30 years' worth of telegraph line infrastructure in place from coast to coast and in most major cities throughout the United States. The initial telephone services used existing telegraph links to distribute telegrams, but this was not a satisfactory solution. The telegraph network was optimized for the dots and dashes of Morse code, not the smooth analog waveforms of voice. Early attempts to run voice over telegraph lines stumbled not over bandwidth, but with the crosstalk induced

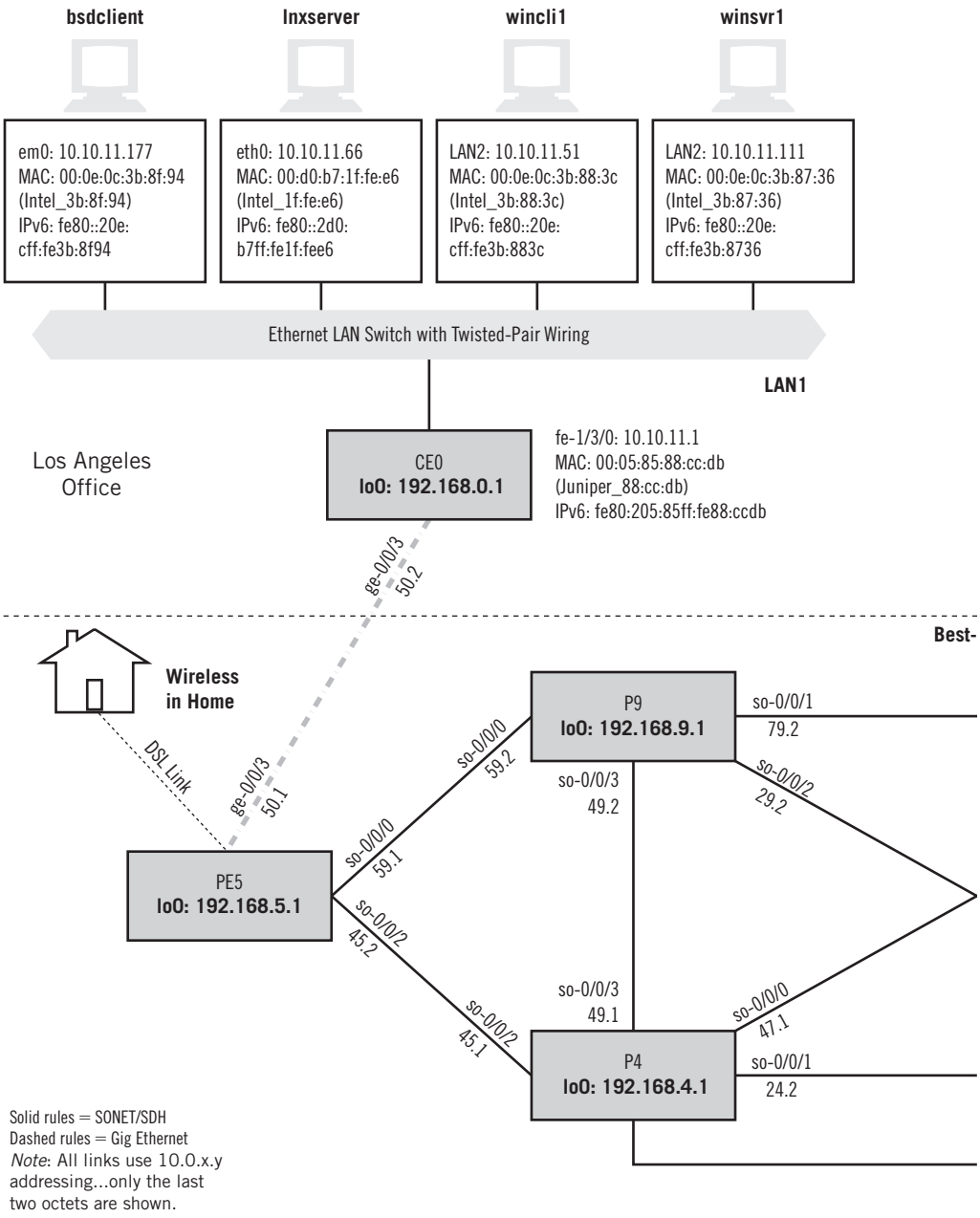
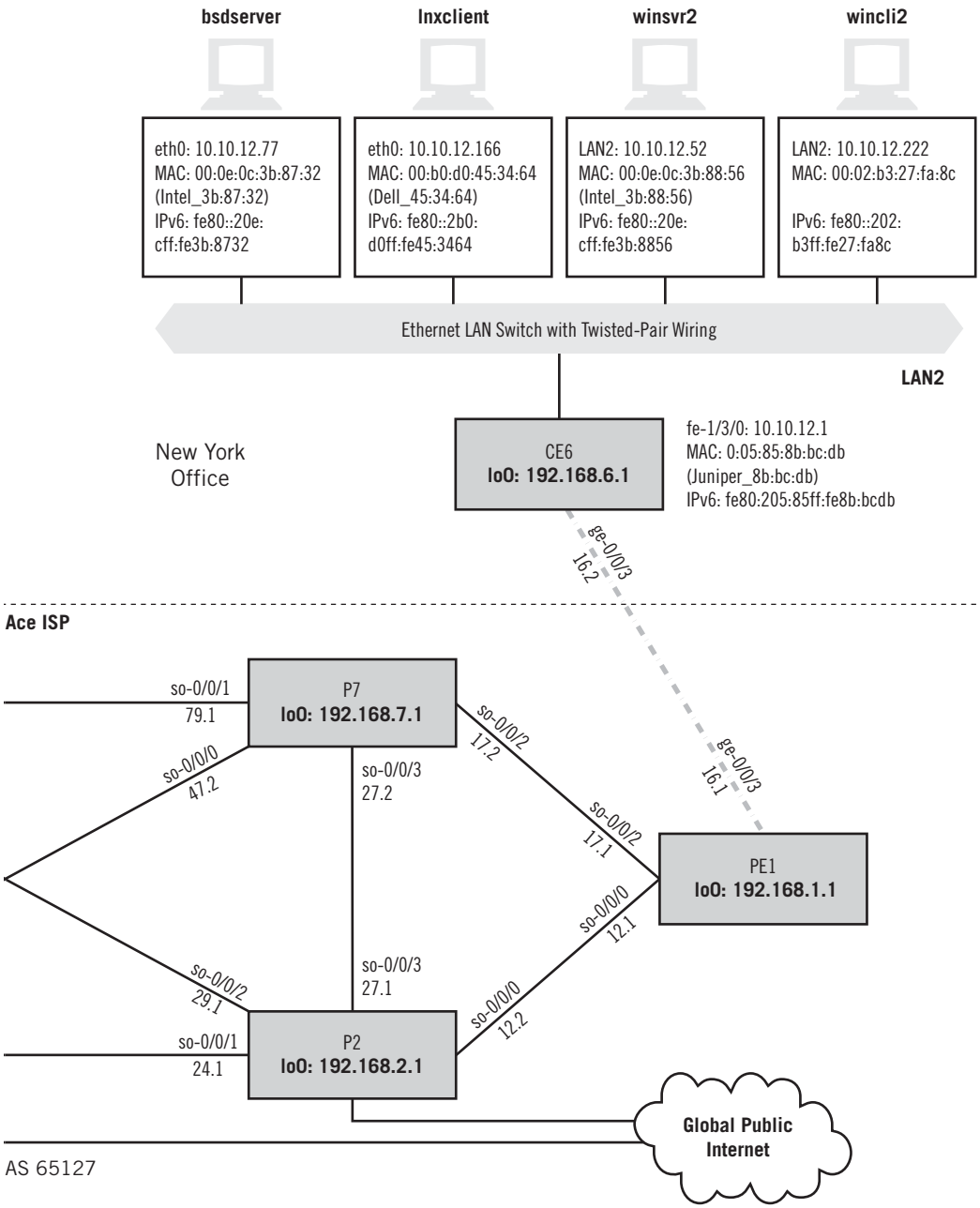


FIGURE 17.1

The routers on the Illustrated Network will be used to illustrate MPLS. Note that we are still dealing with the merged Best-Ace ISP and a single AS.


bsdserver
lnxclient
winsvr2
wincli2
eth0: 10.10.12.77
MAC: 00:0e:0c:3b:87:32
(Intel_3b:87:32)
IPv6: fe80::20e:
cff:fe3b:8732
eth0: 10.10.12.166
MAC: 00:b0:d0:45:34:64
(Dell_45:34:64)
IPv6: fe80::2b0:
d0ff:fe45:3464
LAN2: 10.10.12.52
MAC: 00:0e:0c:3b:88:56
(Intel_3b:88:56)
IPv6: fe80::20e:
cff:fe3b:8856
LAN2: 10.10.12.222
MAC: 00:02:b3:27:fa:8c
IPv6: fe80::202:
b3ff:fe27:fa8c
Ethernet LAN Switch with Twisted-Pair Wiring
LAN2
New York
Office
CE6
lo0: 192.168.6.1
fe-1/3/0: 10.10.12.1
MAC: 0:05:85:8b:bc:db
(Juniper_8b:bc:db)
IPv6: fe80:205:85ff:fe8b:bcdb
ge-0/0/3
16.2
Ace ISP
so-0/0/1
79.1
P7
lo0: 192.168.7.1
so-0/0/2
17.2
so-0/0/0
47.2
so-0/0/3
27.2
so-0/0/2
17.1
ge-0/0/3
16.1
PE1
lo0: 192.168.1.1
so-0/0/0
12.1
so-0/0/2
29.1
so-0/0/3
27.1
so-0/0/0
12.2
so-0/0/1
24.1
P2
lo0: 192.168.2.1
Global Public
Internet
AS 65127

by the pulses of Morse code running in adjacent wires. The solution was to twist and pair telephone wires and maintain adequate separation from telegraph wire bundles.

So, two separate networks grew up: telephone and telegraph. When cable TV came along much later, the inadequate bandwidth of twisted-pair wire led to a third major distinct network architecture—this one made of coaxial cable capable of delivering 50 or more (compared to the handful of broadcast channels available, that was a lot) television channels at the same time.

Naturally, communications companies did not want to pay for, deploy, and maintain three separate networks for separate services. It was much more efficient to use one converged infrastructure for everything. Once deregulation came to the telecommunications industry, and the same corporate entity could deliver voice as a telephony company, video as a cable TV company, and data as an ISP, the pressure to find a "universal" network architecture became intense. But the Internet was not the only universal network intended to be used for the convergence of voice, video, and data over the same links. Telecommunications companies also used frame relay (FR) and asynchronous transfer mode (ATM) networks to try to carry voice, video, and data on the same links.

Let's see if we can "converge" these different applications onto the Illustrated Network. This chapter will use the Illustrated Network routers exclusively. This is shown in Figure 17.1, which also reveals something interesting when we run traceroute from `bsdclient` on LAN1 to `bsdserver` on LAN2.

```
bsdclient# traceroute bsdserver

traceroute to bsdserver (10.10.12.77), 64 hops max, 44 byte packets
 1  10.10.11.1 (10.10.11.1)  0.363 ms  0.306 ms  0.345 ms
 2  10.0.50.1 (10.1.36.2)  0.329 ms  0.342 ms  0.346 ms
 3  10.0.45.1 (10.0.45.1) 0.330 ms  0.341 ms  0.346 ms
 4  10.0.24.1 (10.0.24.1) 0.332 ms  0.343 ms  0.345 ms
 5  10.0.12.1 (10.0.12.1) 0.329 ms  0.342 ms  0.347 ms
 6  10.0.16.2 (10.0.16.2) 0.330 ms  0.341 ms  0.346 ms
 7  10.10.12.77 (10.10.12.77)  0.331 ms  0.343 ms  0.347 ms

bsdclient#
```

The packets travel from PE5 to P4 and then on to P2 and PE1. Why shouldn't they flow through P9 and P7? Well, they could, but without load balancing turned on (and it is not) PE5 has to choose P9 or P4 as the next hop. All things being equal, if all other metrics are the same, routers typically pick to lowest IP address. A look at the network diagram shows this to be the case here.

There are obviously other users on the Best-Ace ISP's network, not just those on LAN1 and LAN2. However, it would be nice if the customer-edge (site) routers CE0 and CE6 were always seven hops away and never any more (in other words, no matter how traffic is routed there are always six routers between LAN1 and LAN2). This is because most of the traffic flows between the two sites, as we have seen (on many LANs, vast quantities of traffic usually flow among a handful of destinations).

Before the rise of the Internet, the company owning LAN1 and LAN2 would pay a service provider (telephone company or other "common carrier") to run a point-to-point

link between New York and Los Angeles and use it for data traffic. They might also do the same for voice, and perhaps even for video conferences between the two sites. The nice thing about these leased line links (links used exclusively for voice are called tie lines) is that they make the two sites appear to be directly connected, reducing the number of hops (and network processing delay) drastically.

But leased lines are an expensive solution (they are paid for by the mile) and are limited in application (they only connect the two sites). What else could a public network service provider offer as a convergence solution to make the network more efficient?

We'll take a very brief look at the ideas behind some public network attempts at convergence (frame relay and ATM) and then see how TCP/IP itself handles the issue. We'll introduce Multiprotocol Label Switching (MPLS) and position this technology as a way to make IP router networks run faster and more efficiently with IP switching.

CONVERGING WHAT?

Convergence is not physical convergence through channels, which had been done for a very long time. Consider a transport network composed of a series of fiber optic links between SONET/SDH multiplexers. The enormous bandwidth on these links can be (and frequently is) channelized into multiple separate paths for voice bits, data bits, and video bits on the same physical fiber. But this is not convergence.

In this chapter convergence means the combination of voice, video, and data on the same physical channel. Convergence means more than just carrying channels on the same physical transport. It means combining the bits representing voice, video, and data into one stream and carrying them all over the total bandwidth on the same "unchannelized" fiber optic link. If there are voice, video, and data channels on the link, these are now virtual channels (or logical channels) and originate and terminate in the same equipment—not only at the physical layer, but at some layer above the lowest.

On modern Metro Ethernet links, the convergence is done by combining the traffic from separate VLANs on the same physical transport. The VLANs can be established based on traffic type (voice, video, and data), customer or customer site, or both (with an inner and outer VLAN label.) In this chapter, we'll talk about MPLS—which can work with VLANs or virtual channels.

Fast Packet Switching

Before there was MPLS, there was the concept of *fast packet switching* to speed up packet forwarding on converged links and through Internet network nodes. Two major technologies were developed to address this new technology, and they are worth at least a mention because they still exist in some places.

Frame Relay

Frame relay was an attempt to slim down the bulky X.25 public packet switching standard protocol stack for public packet networks for the new environment of home PCs and computers at every work location in an organization. Although it predated

modern layered concepts, X.25 essentially defined the data units at the bottom three layers—physical interface, frame structure, and packet—as an international standard. It was mildly successful compared to the Internet, but wildly successful for a world without the Web and satellite or cell phones. In the mid-1980s, about the only way to communicate text to an off-shore oil platform or ships at sea was with the familiar but terse "GA" (go ahead) greeting on a teletype over an X.25 connection.

The problem with X.25 packets (called PLP, Packet Layer Protocol, packets) was that they weren't IP packets, and so could not easily share or even interface with the Internet, which had started to take off when the PC hit town. But IP didn't have a popular WAN frame defined (SLIP did not really use frames), so the X.25 Layer 2 frame structure, High-level Data Link Control (HDLC)—also used in ISDN—was modified to make it more useful in an IP environment populated by routers. In fact, routers, which struggled with full X.25 interfaces, could easily add frame relay interfaces.

One of the biggest parts of X.25 dropped on the way to frame relay was error resistance. Today, network experts have a more nuanced and sophisticated understanding of how this should be done instead of the heavyweight X.25 approach to error detection and recovery.

Frame relay was once popularly known as "X.25 on steroids," a choice of analogies that proved unfortunate for both X.25 and frame relay. But at least frame relay switch network nodes could relay frames faster than X.25 switches could route packets. Attempts were made to speed X.25 up prior to the frame relay makeover, such as allowing a connection-request message to carry data, which was then processed and a reply returned by the destination in a connection-rejected message, thus making X.25 networks as efficient for some things as a TCP/IP network with UDP. However, an X.25 network was still much more costly to build and operate than anything based on the simple Internet architecture. The optimization to X.25 that frame relay represented is shown in Figure 17.2.

Even with frame relay defined, there was still one nagging problem: Like X.25 before it, frame relay was connection oriented. Only signaling protocol messages were connectionless, and many frame relay networks used "permanent virtual circuits" set up

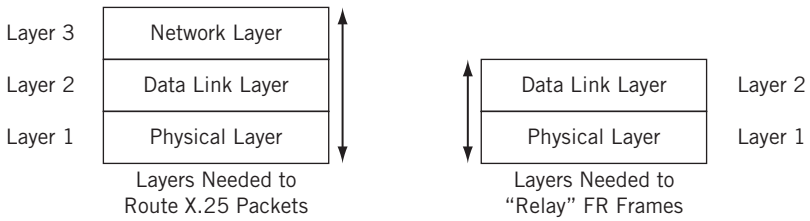


FIGURE 17.2

How X.25 packet routing relates to frame relaying. Note that frame relay has no network layer, leaving IP free to function independently.

with a labor-intensive process comparable to configuring router tables with hundreds of static entries in the absence of mature routing protocols.

Connections were a large part of the reason that X.25 network nodes were switches and not routers. A network node that handled only frame relay frames was still a switch, and connections were now defined by a simple identifier in the frame relay header and called "virtual circuits." But a connection was still a connection. In the time it took a frame relay signaling message exchange to set up a connection, IP with UDP could send a request and receive a reply. Even for bulk data transfer, connections over frame relay had few attractions compared to TCP for IP.

The frame relay frame itself was tailor-made for transporting IP packets over public data networks run by large telecommunications carriers rather than privately owned routers linked by dedicated bandwidth leased by the mile from these same carriers. The frame relay frame structure is shown in Figure 17.3.

- *DLCI*—The Data Link Connection Identifier is a 10-bit field that gives the connection number.
- *C/R*—The Command/Response bit is inherited from X.25 and not used.
- *EA*—The Extended Address bit tells whether the byte is the last in the header (headers in frame relay can be longer than 2 bytes).
- *FECN and BECN*—The Forward/Backward Explicit Congestion Notification bits are used for flow control.
- *DE*—The Discard Eligible bit is used to identify frames to discard under congested conditions.

Unlike a connectionless packet, the frame relay frame needs only a connection identifier to allow network switch nodes to route the frame. In frame relay, this is the DLCI. A connection by definition links two hosts, source and destination. There is no

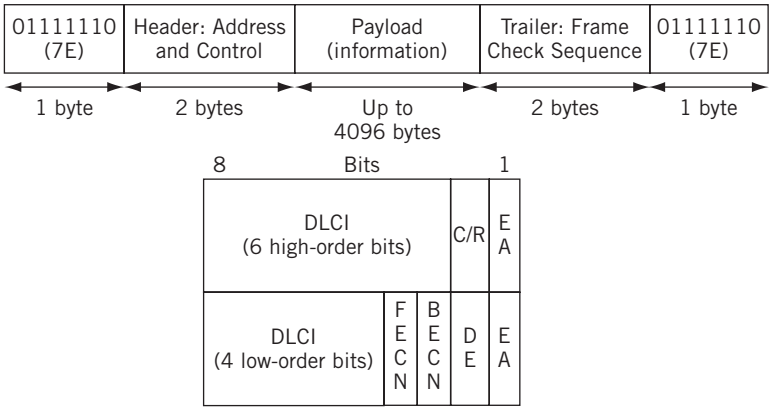


FIGURE 17.3

The basic 2-byte frame relay frame and header. The DLCI field can come in larger sizes.

sense of "send this *to* DLCI 18" or "this is *from* DLCI 18." Frames travel *on* DLCI 18, and this implies that connections are inherently unidirectional (which they are, but are usually set up and released in pairs) and that the connection identifiers in each direction did not have to match (although they typically did, just to keep network operators sane).

One of the things that complicate DLCI discussions is that unlike globally unique IP addresses, DLCIs have local significance only. This just means that the DLCI on a frame relay frame sent from site A on DLCI 25 could easily arrive at site B on DLCI 38. And in between, the frame could have been passed around the switches as DLCI 18, 44, or whatever. Site A only needs to know that the local DLCI 25 leads to site B, and site B needs to know that DLCI 38 leads to site A, and the entire scheme still works. But it is somewhat jarring to TCP/IP veterans.

This limits the connectivity from each site to the number of unique DLCIs that can operate at any one time, but the DLCI header field can grow if this becomes a problem. And frame relay connections were never supposed to be used all of the time.

What about adding voice and video to frame relay? That was actually done, especially with voice. Frame relay was positioned as a less expensive way of linking an organization's private voice switches (called private branch exchanges, or PBXs) than with private voice circuits. Voice was not always packetized, but at least it was "framerized" over these links. If the links had enough bandwidth, which was not always a given, primitive videoconferencing (but not commercial-quality video signals that anyone would pay to view) could be used as well.

Frame relay suffered from three problems, which proved insurmountable. It was not particularly IP friendly, so frame relay switches (which did not run normal IP routing protocols) could not react to TCP/IP network conditions the way routers could. The router and switches remained "invisible" to each other. And in spite of efforts to integrate voice and video onto the data network, frame relay was first and foremost a data service and addressed voice and video delay concerns by grossly overconfiguring bandwidth in almost all cases. Finally, the telecommunications carriers (unlike the ISPs) resisted easy interconnection of the frame relay network with those of other carriers, which forced even otherwise eager customers to try to do everything with one carrier (an often impossible task). It was a little like cell phones without any possibility of roaming, and in ironic contrast to the carrier's own behavior as an ISP, this closed environment was not what customers wanted or needed.

Frame relay still exists as a service offering. However, outside of just another type of router WAN interface, frame relay has little impact on the Internet or IP world.

Asynchronous Transfer Mode

The Asynchronous Transfer Mode (ATM) was the most ambitious of all convergence methods. It had to be, because what ATM essentially proposed was to throw everything out that had come before and to "Greenfield" the entire telecommunications structure

the world over. ATM was part of an all-encompassing vision of networking known as broadband ISDN (B-ISDN), which would support all types of voice, video, and data applications though virtual channels (and virtual connections). In this model, the Internet would yield to a global B-ISDN network—and TCP/IP to ATM.

Does this support plan for converged information sound familiar? Of course it does. It's pretty much what the Internet and TCP/IP do today, without B-ISDN or ATM. But when ATM was first proposed, the Internet and TCP/IP could do none of the things that ATM was supposed to do with ease. How did ATM handle the problems of mixing support for bulk data transfer with the needs of delay-sensitive voice and bandwidth-hungry (and delay-sensitive) video?

ATM was the international standard for what was known as cell relay (there were cell relay technologies other than ATM, now mostly forgotten). The cell relay name seems to have developed out of an analogy with frame relay. Frame relay "relayed" (switched) Layer 2 frames through network nodes instead of independently routing Layer 3 packets. The efficiency of doing it all at a lower layer made the frame relay node faster than a router could have been at the time.

Cell relay took it a step further, doing everything at Layer 1 (the actual bit level). But there was no natural data unit at the physical layer, just a stream of bits. So, they invented one 53 bytes long and called it the "cell"—apparently in comparison to the cell in the human body—which is very small, can be generic, and everything else is built up from them. Technically, in data protocol stacks, cells are a "shim" layer slipped between the bits and the frames, because both bits and frames are still needed in hardware and software at source and destination.

Cell relay (ATM) "relayed" (switched) cells through network nodes. This could be done entirely in hardware because cells were all exactly the same size. Imagine how fast ATM switches would be compared to slow Layer 3 routers with two more layers to deal with! And ATM switches had no need to allocate buffers in variable units, or to clean up fragmented memory. The structure of the 5-byte ATM cell header is shown in Figure 17.4 (descriptions follow on next page). The call payload is always 48 bytes long.

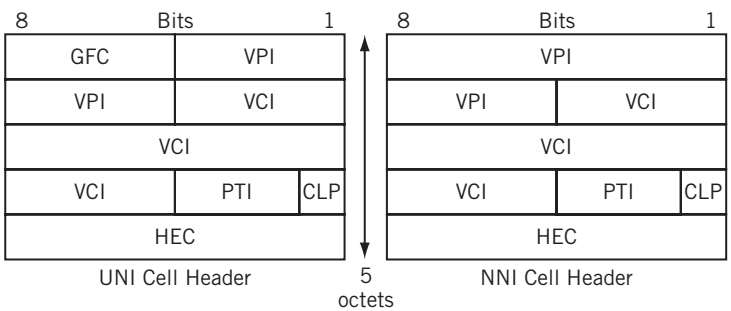


FIGURE 17.4

The ATM cell header. Note the larger VPI fields on the network (NNI) version of the header.

- *GFC*—The Generic Flow Control is a 4-bit field used between a customer site and ATM switch, on the User-Network Interface (UNI). It is not present on the Network–Network Interface (NNI) between ATM switches.
- *VPI*—The Virtual Path Identifier is an 8- or 12-bit field used to identify paths between sites on the ATM network. It is larger on the NNI to accommodate aggregation on customer paths.
- *VCI*—The Virtual Connection Identifier is a 16-bit field used to identify paths between individual devices on the ATM network.
- *PTI*—The Payload Type Indicator is a 3-bit field used to identify one of eight traffic types carried in the cell.
- *CLP*—The Cell Loss Priority bit serves the same function as the DE bit in frame relay, but identifies cells to discard when congestion occurs.
- *HEC*—The Header Error Control byte not only detects bit errors in the entire 40-bit header, but can also *correct* single bit errors.

In contrast to frame relay, the ATM connection identifier was a two-part virtual path identifier (VPI) and virtual channel identifier (VCI). Loosely, VPIs were for connections between sites and VCIs were for connections between devices. ATM switches could "route" cells based on the VPI, and the local ATM switch could take care of finding the exact device for which the cell was destined.

Like frame relay DLCIs, ATM VPI/VCIs have local significance only. That is, the VPI/VPI values change as the cells make their way from switch to switch and depending on direction. Both frame relay and ATM switch essentially take a data unit in on an input port, look up the header (DLCI or VPI/VCI label) in a table, and output the data unit on the port indicated in the table—but also with a new label value, also provided by the table.

This distinctive label-swapping is characteristic of switching technologies and protocols. And, as we will see later, switching has come to the IP world with MPLS, which takes the best of frame relay and ATM and applies it directly to IP without the burden of "legacy" stacks (frame relay) or phantom applications (ATM and B-ISDN).

The tiny 48-byte payload of the ATM cell was intentional. It made sure that no delay-sensitive bits got stuck in a queue behind some monstrous chunk of data a thousand times larger than the 48 voice or video bytes. Such "serialization delay" introduced added delay and delay variation (jitter) that rendered converged voice and video almost useless without more bandwidth than anyone could realistically afford. With ATM, all data encountered was a slightly elevated delay when data cells shared the total bandwidth with voice and video. But because few applications did anything with data (such as a file) before the entire group of bits was transferred intact ATM pioneers deemed this a minor inconvenience at worst.

All of this sounded too good to be true to a lot of networking people, and it turned out that it was. The problem was not with raw voice and video, which could be molded into any form necessary for transport across a network. The issue was with data, which came inside IP packets and had to be broken down into 48-byte units—each of which had a 5-byte ATM cell header, and often a footer that limited it to only 30 bytes.

This was an enormous amount of overhead for data applications, which normally added 3 or 4 bytes to an Ethernet frame for transport across a WAN. Naturally, no hardware existed to convert data frames to cells and back—and software was much too slow—so this equipment had to be invented. Early results seemed promising, although the frame-to-cell-and-back process was much more complex and expensive than anticipated. But after ATM caught on, prices would drop and efficiencies would be naturally discovered. Once ATM networks were deployed, the B-ISDN applications that made the most of them would appear. Or so it seemed.

However, by the early 1990s it turned out that making cells out of data frames was effective as long as the bandwidth on the link used to carry both voice and video along with the data was limited to less than that needed to carry all three at once. In other words, if the link was limited to 50 Mbps and the voice and video data added up to 75 Mbps, cells made sense. Otherwise, variable-length data units worked just fine. Full-motion video was the killer at the time, with most television signals needing about 45 Mbps (and this was not even high-definition TV). Not only that, but it turned out that the point of diminishing ATM returns (the link bandwidth at which it became slower and more costly to make cells than simply send variable-length data units) was about 622 Mbps—lower than most had anticipated.

Of course, one major legacy of the Internet bubble was the underutilization of fiber optic links with more than 45 Mbps, and in many cases greatly in excess of 622 Mbps. And digital video could produce stunning images with less and less bandwidth as time went on. And in that world, in many cases, ATM was left as a solution without a problem. ATM did not suffer from lack of supporters, but it proved to be the wrong technology to carry forward as a switching technology for IP networks.

Why Converge on TCP/IP?

Some of the general reasons TCP/IP has dominated the networking scene have been mentioned in earlier chapters. Specifically, none of the "new" public network technologies were particularly TCP/IP friendly—and some seemed almost antagonistic. ATM cells, for instance, would be a lot more TCP/IP friendly if the payload were 64 bytes instead of 48 bytes. At least a lot of TCP/IP traffic would fit inside a single ATM cell intact, making processing straightforward and efficient.

At 48 bytes, everything in TCP/IP had to be broken up into at least two cells. But the voice people wanted the cell to be 32 bytes or smaller, in order to keep voice delays as short as possible. It may be only a coincidence that 48 bytes is halfway between 32 and 64 bytes, but a lot of times reaching a compromise instead of making a decision annoys both parties and leaves neither satisfied with the result. So, ATM began as a standard by alienating the two groups (voice and data) that were absolutely necessary to make ATM a success.

But the real blow to ATM came because a lot of TCP/IP traffic would not fit into 64-byte frames. ACKs would fit well, but TCP/IP packet sizes tend to follow a bimodal distribution with two distinct peaks at about 64 and between 1210 and 1550 bytes. The upper cluster is smaller and more spread out, but this represents the vast bulk of all traffic on the Internet.

Then new architectures allowed otherwise normal IP routers to act like frame relay and ATM switches with the addition of IP-centric MPLS. Suddenly, all of the benefits of frame relay and ATM could be had without using unfamiliar and special equipment (although a router upgrade might be called for).

MPLS

Rather than adding IP to fast packet switching networks, such as frame relay and ATM, MPLS adds fast packet switching to IP router networks. We've already talked about some of the differences between routing (connectionless networks) and switching networks in Chapter 13. Table 17.1 makes the same type of comparisons from a different perspective.

The difference in the way CoS is handled is the major issue when convergence is concerned. Naturally, the problem is to find the voice and video packets in the midst of the data packets and make sure that delay-sensitive packets are not fighting for bandwidth along with bulk file transfers or email. This is challenging in IP routers because there is no fixed path set up through the network to make it easy to enforce QoS at every hop along the way. But switching uses stable paths, which makes it easy to determine exactly which routers and resources are consumed by the packet stream. QoS is also challenging because you don't have administrative control over the routers outside your own domain.

MPLS and Tunnels

Some observers do not apply the term "tunnel" to MPLS at all. They reserve the term for wholesale violations on normal encapsulations (packet in frame in a packet, for example). MPLS uses a special header (sometimes called a "shim" header) between packet and frame header, a header that is not part of the usual TCP/IP suite layers.

However, RFCs (such as RFC 2547 and 4364) apply the tunnel terminology to MPLS. MPLS headers certainly conform to general tunnel "rules" about stack encapsulation violations. This chapter will not dwell on "MPLS tunnel" terminology but will not avoid the term either. (This note also applies to MPLS-based VPNs, discussed in Chapter 26.)

But QoS enforcement is not the only attraction of MPLS. There are at least two others, and probably more. One is the ability to do *traffic engineering* with MPLS, and the other is that MPLS *tunnels* form the basis for a certain *virtual private network* (VPN) scheme called *Layer 3 VPNs*. There are also *Layer 2 VPNs*, and we'll look at them in more detail in Chapter 26.

MPLS uses tunnels in the generic sense: The normal flow of the layers is altered at one point or another, typically by the insertion of an "extra" header. This header is added at one end router and removed (and processed) at the other end. In MPLS, routers form the

Table 17.1 Comparing Routing and Switching on a WAN

Characteristic	Routing	Switching
Network node	Router	Switch
Traffic flow	Each packet routed independently hop by hop	Each data unit follows same path through network
Node coordination	Routing protocols share information	Signaling protocols set up paths through network
Addressing	Global, unique	Label, local significance
Consistency of address	Unchanged source to destination	Label is swapped at each node
QoS	Challenging	Associated with path

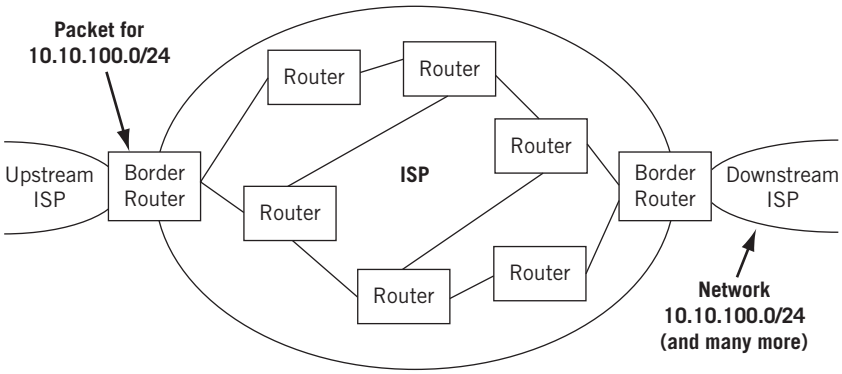


FIGURE 17.5

The rationale for MPLS. The LSP forms a "shortcut" across the routing network for transit traffic. The Border Router knows right away, thanks to BGP, that the packet for 10.10.100.0/24 must exit at the other border router. Why route it independently at every router in between?

endpoints of the tunnels. In MPLS, the header is called a *label* and is placed between the IP header and the frame headers—making MPLS a kind of "Layer 2 and a half" protocol.

MPLS did not start out to be the answer to everyone's dream for convergence or traffic engineering or anything else. MPLS addressed a simple problem faced by every large ISP in the world, a problem shown in Figure 17.5.

MPLS was conceived as a sort of BGP "shortcut" connecting border routers across the ISP. As shown in the figure, a packet bound for `10.10.100.0/24` entering the border router from the upstream ISP is known, thanks to the IBGP information, to have to exit the ISP at the other border router. In practice, of course, this will apply to many border routers and thousands of routes (usually most of them), but the principle is the same.

Only the local packets with destinations within the ISP technically need to be routed by the interior routers. Transit packets can be sent directly to the border router,

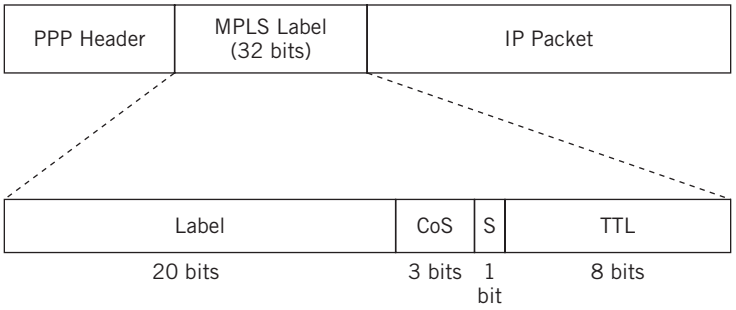


FIGURE 17.6

The 32-bit MPLS label fields. Note the 3-bit CoS field, which is often related to the IP ToS header. The label field is used to identify flows that should be kept together as they cross the network.

if possible. MPLS provides this mechanism, which works with BGP to set up tunnels through the ISP between the border routers (or anywhere else the ISP decides to use them).

The structure of the label used in MPLS is shown in Figure 17.6. In the figure, it is shown between a Layer 2 PPP frame and the Layer 3 IP packet (which is very common).

- *Label*—This 20-bit field identifies the packets included in the "flow" through the MPLS tunnel.
- *CoS*—Class-of-Service is a 3-bit field used to classify the data stream into one of eight categories.
- *S*—The Stack bit lets the router know if another label is *stacked* after the current 32-bit label.
- *TTL*—The Time-to-Live is an 8-bit field used in exactly the same way as the IP packet header TTL. This value can be copied from or into the IP packet or used in other ways.

Certain label values and ranges have been reserved for MPLS. These are outlined in Table 17.2.

The MPLS architecture is defined in RFC 3031, and MPLS label stacking is defined in RFC 3032 (more than one MPLS label can precede an IP packet). General traffic engineering in MPLS is described in RFC 2702, and several drafts add details and features to these basics.

What does it mean to use traffic engineering on a router network? Consider the Illustrated Network. We saw that traffic from LAN1 to LAN2 flows through backbone routers P4 and P2 (reverse traffic also flows this way). But notice that P2 and P4 also have links to and from the Internet. A lot of general Internet traffic flows through routers P2 and P4 and their links, as well as LAN1 and LAN2 traffic.

Table 17.2 MPLS Label Values and Their Uses

Value or Range	Use
0	IPv4 Explicit Null. Must be the last label (no stacking). Receiver removes the label and routes the IPv4 packet inside.
1	Router Alert. The IP packet inside has information for the router itself, and the packet should not be forwarded.
2	IPv6 Explicit Null. Same as label 0, but with IPv6 inside.
3	Implicit Null. A "virtual" label that never appears in the label itself. It is a table entry to request label removal by the downstream router.
4–15	Reserved.
16–1023 and 10000–99999	Ranges used in Juniper Networks routers to manually configure MPLS tunnels (not used by the signaling protocols).
1024–9999	Reserved.
100000–1048575	Used by signaling protocols.

So, it would make sense to "split off" the LAN1 and LAN2 traffic onto a less utilized path through the network (for example, from PE5 to P9 to P7 to PE1). This will ease congestion and might even be faster, even though in some configurations there might be more hops (for example, there might be other routers between P9 and P7).

Why Not Include CE0 and CE6?

Why did we start the MPLS tunnels at the provider-edge routers instead of directly at the customer edge, on the premises? Actually, as long as the (generally) smaller site routers support the full suite of MPLS features and protocols there's no reason the tunnel could not span LAN to LAN.

However, MPLS traditionally begins and ends in the "provider cloud"—usually on the PE routers, as in this chapter. This allows the customer routers to be more independent and less costly, and allows reconfiguration of MPLS without access to the customer's routers. Of course, in some cases the customer might want ISP to handle MPLS management—and then the CE routers certainly could be included on the MPLS path.

There are ways to do this with IGPs, such as OSPF and IS-IS, by adjusting the link metrics, but these solutions are not absolute and have global effects on the network. In contrast, an MPLS tunnel can be configured from PE5 to PE1 through P9 and P7 and

only affect the routing on PE5 and PE1 that involves LAN1 and LAN2 traffic, exactly the effect that is desired.

MPLS Terminology

Before looking at how MPLS would handle a packet sent from LAN1 to LAN2 over an MPLS tunnel, we should look at the special terminology involved with MPLS. In no particular order, the important terms are:

LSP—We've been calling them tunnels, and they are, but in MPLS the tunnel is called a *label-switched path*. The LSP is a unidirectional connection following the same path through the network.

Ingress router—The *ingress router* is the start of the LSP and where the label is *pushed* onto the packet.

Egress router—The *egress router* is the end of the LSP and where the label is *popped* off the packet.

Transit or intermediate router—There must be at least one *transit* (sometimes called *intermediate*) *router* between ingress and egress routers. The transit router(s) *swaps* labels and replaces the incoming values with the outgoing values.

Static LSPs—These are LSPs set up by hand, much like *permanent virtual circuits* (PVCs) in FR and ATM. They are difficult to change rapidly.

Signaled LSPs—These are LSPs set up by a signaling protocol used with MPLS (there are two) and are similar to *switched-virtual circuits* (SVCs) in FR and ATM.

MPLS domain—The collection of routers within a routing domain that starts and ends all LSPs form the MPLS domain. MPLS domains can be nested, and can be a subset of the routing domain itself (that is, all routers do not have to understand MPLS; only those on the LSP).

Push, pop, and swap—A *push* adds a label to an IP packet or another MPLS label. A *pop* removes and processes a label from an IP packet or another MPLS label. A *swap* is a pop followed by a push and replaces one label by another (with different field values). Multiple labels can be added (push push . . .) or removed (pop pop . . .) at the same time.

Penultimate hop popping (PHP)—Many of LSPs can terminate at the same border router. This router must not only pop and process all the labels but route all packets inside, *plus* all other packets that arrive from within the ISP. To ease the load of this border router, the router one hop upstream from the egress router (known as the *penultimate* router) can pop the label and simply route the packet to the egress router (it *must* be one hop, so the effect is the

same). PHP is an optional feature of LSPs, and keep in mind that the LSP is still considered to terminate at the egress router (not at the penultimate).

Constrained path LSPs—These are *traffic engineering* (TE) LSPs set up by a signaling protocol that must respect certain TE constraints imposed on the network with regard to delay, security, and so on. TE is the most intriguing aspect of MPLS.

IGP shortcuts—Usually, LSPs are used in special router tables and only available to routes learned by BGP (transit traffic). Interior Gateway Protocol (IGP) shortcuts allow LSPs to be installed in the main routing table and used by traffic within the ISP itself, routes learned by OSPF or another IGP.

Signaling and MPLS

There are two signaling protocols that can be used in MPLS to automatically set up LSPs without human intervention (other than configuring the signaling protocols themselves!). The Resource Reservation Protocol (RSVP) was originally invented to set up QoS "paths" from host to host through a router network, but it never scaled well or worked as advertised. Today, RSVP has been defined in RFC 3209 as RSVP for TE and is used as a signaling protocol for MPLS. RSVP is used almost exclusively as RSVP-TE (most people just say RSVP) by routers to set up LSPs (explicit-path LSPs), but can still be used for QoS purposes (constrained-path LSPs).

The Label Distribution Protocol (LDP), defined in RFC 3212, is used exclusively with MPLS but cannot be used for adding QoS to LSPs other than using simple constraints when setting up paths. On the other hand, LDP is trivial to configure compared to RSVP. This is because LDP works directly from the tables created by the IGP (OSPF or IS–IS). The lack of QoS support in LDP is due to the lack of any intention in the process. The reason for the LDP paths created from the IGP table to exist is only simple adjacency. In addition, LDP does not offer much if your routing platform can forward packets almost as fast as it can switch labels. Today, use of LDP is deprecated (see the admonitions in RFC 3468) in favor of RSVP-TE.

A lot of TCP/IP texts spend a lot of time explaining how RSVP-TE works (they deal with LDP less often). This is more of an artifact of the original use of RSVP as a host-based protocol. It is enough to note that RSVP messages are exchanged between all routers along the LSP from ingress to egress. The LSP label values are determined, and TE constraints respected, hop by hop through the network until the LSP is ready for traffic. The process is quick and efficient, but there are few parameters that can be configured even on routers that change RSVP operation significantly (such as interval timers)—and none at all on hosts.

Although not discussed in detail in this introduction to MPLS, another protocol is commonly used for MPLS signaling, as described in RFC 2547bis. BGP is a routing protocol, not a signaling protocol, but the extensions used in multiprotocol BPG (MPBGP) make it well suited for the types of path setup tasks described in this chapter. With MPBGP, it is possible to deploy BGP- and MPLS-based VPNs without the use of any other

signaling protocol. LSPs are established based on the routing information distributed by MPBGP from PE to PE. MPBGP is backward compatible with "normal" BGP, and thus use of these extensions does not require a wholesale upgrade of all routers at once.

Label Stacking

Of all the MPLS terms outlined in the previous section, the one that is essential to understand is the concept of "nested" LSPs; that is, LSPs which include one or more other LSPs along their path from ingress to egress. When this happens, there will be more than one label in front of the IP packet for at least part of its journey.

It is common for many large ISPs to stack three labels in front of an IP packet. Often, the end of two LSPs is at the same router and two labels are pushed or popped at once. The current limit is eight labels.

There are several instances where this stacking ability comes in handy. A larger ISP can buy a smaller ISP and simply "add" their own LSPs onto (outside) the existing ones. In addition, when different signaling protocols are used in core routers and border routers, these domains can be nested instead of discarding one or the other.

The general idea of nested MPLS domains with label stacking is shown in Figure 17.7. There are five MPLS domains, each with its own way of setting up LSPs: static, RSVP, and LDP. The figure shows the number of labels stacked at each point and the order

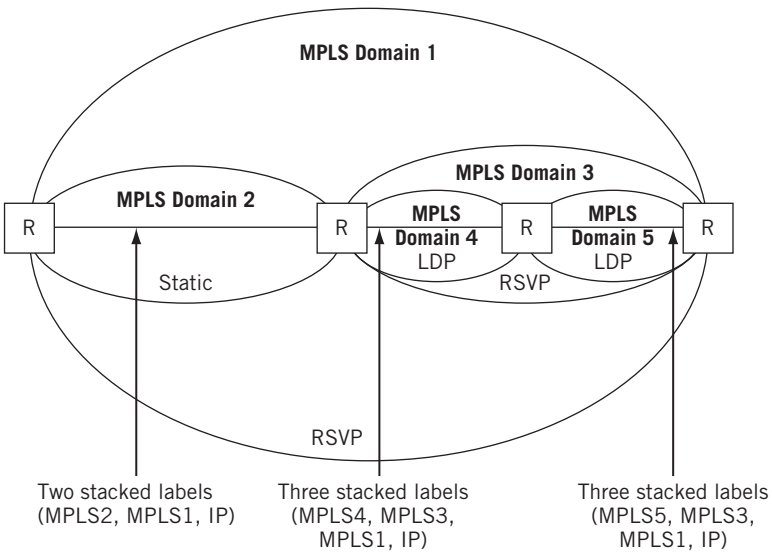


FIGURE 17.7

MPLS domains, showing how the domains can be nested or chained, and how multiple labels are used.

they are stacked in front of the packet. All of the routers shown (in practice, there will be many more) pop and process multiple labels. MPLS domains can be nested for geographical, vendor, or organizational reasons as well.

MPLS and VPNs

MPLS forms the basis for many types of VPNs used on IP networks today, especially Layer 3 VPNs. LSPs are like the PVCs and SVCs that formed "virtually private" links across a shared public network such as FR or ATM. LSPs are not really the same as private leased-line links, but they appear to be to their users.

Of course, while the path is constrained, the MPLS-based Layer 3 VPN is not actually doing anything special to secure the content of the tunnel or to protect its integrity. So, this "security" value is limited to constraining the path. This reduces the places where snooping or injection can occur, but it does not replace other Layer 3 VPN technology for security (such as IPSec, discussed in Chapter 29).

Nevertheless, VPNs are often positioned as a security feature on router networks. This is because, like "private" circuits, hackers cannot hack into the middle of an LSP (VPN) just by spoofing packets. There are labels to be dealt with, often nested labels. The ingress and egress routers are more vulnerable, but it's not as easy to harm VPNs or the sites they connect as it is to disrupt "straight" router networks.

So, VPNs have a lot in common with MPLS and LSPs—except that the terms are different! For example, the transit routers in MPLS are now provider (P) routers in VPNs. VPNs are discussed further in the security chapters.

MPLS Tables

The tables used to push, pop, and swap labels in multiprotocol label switching are different from the tables used to route packets. This makes sense: MPLS uses switching, and packets are routed.

Most MPLS tables are little more than long lists of labels with two key pieces of information attached: the output interface to the next-hop router on the LSP and the new value of the label. Other pieces of information can be added, but this is the absolute minimum.

What does an MPLS switching table look like? Suppose we did set up an LSP between LAN1 and LAN2 to carry packets from PE5 to PE1 through backbone routers P9 and P7 instead of through P4 and P2?

Figure 17.8 shows how the MPLS switching tables might be set up to switch a packet from LAN1 to LAN2. Note that this has nothing to do with routed traffic going back from LAN2 to LAN1! (In the real world, we would set up an LSP going from LAN2 to LAN1 as well.)

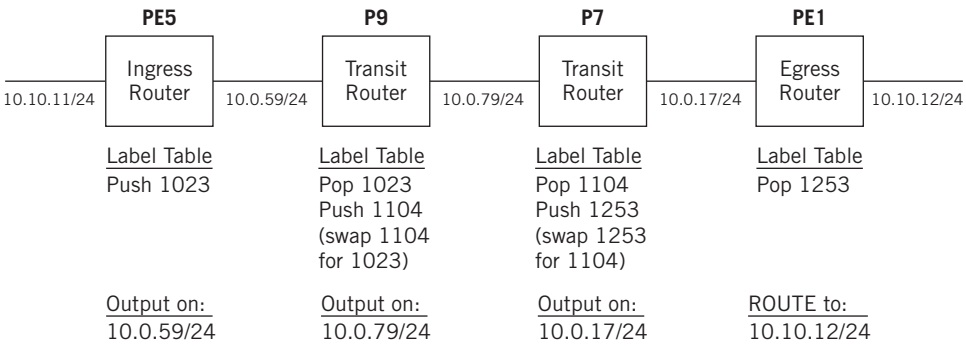


FIGURE 17.8

Label tables for a static LSP from PE5 (ingress) to PE1 (egress).

CONFIGURING MPLS USING STATIC LSPS

Let's build the static LSP from LAN1 to LAN2 from PE5 to P9 to P7 to PE1 that was shown in Figure 17.8. Then we'll show how that affects the routing table entries and run a traceroute for packets sent from `10.10.11.0/24` (LAN1) to `10.10.12.0/24` (LAN2).

The Ingress Router

Let's start by configuring the LSP on PE5, the ingress router, so that packets from LAN1's address space get an MPLS label value of `1023` and are sent to `10.0.59.2` as a next hop on the link to P9 (`so-0/0/0`).

```
set protocols mpls static-path LAN1-to-LAN2 10.10.11.0/24 next-hop 10.0.59.2;
set protocols mpls static-path LAN1-to-LAN2 10.10.11.0/24 push 1023;
set protocols mpls static-path LAN1-to-LAN2 interface so-0/0/0;
```

Once the configuration is committed, the static LSP shows up as a static route naturally (signaled LSPs are referenced by signaling a protocol, RSVP or LDP).

```
admin@PE5# show route table inet.0 protocol static
10.10.11.0/24      *[Static/5]  00:01:42
                   > to 10.0.59.2 via so-0/0/0. push 1023
```

The Transit Routers

This is how the LSP is configured on P9, the first transit (or intermediate) router.

```
set protocols mpls interface so-0/0/0 label-map 1023 next-hop 10.0.79.1;
set protocols mpls interface so-0/0/0 label-map 1023 swap 1104;
```

Note that this table is not organized by destination, as on the PE router, but by the interface that the MPLS data unit arrives on. There can be many labels, but this "label map" looks for 1023, swaps it for label 1104, and forwards it to 10.0.79.1. Note that there was no need to look anything up in the main routing table (in Juniper Networks routers, the interface addresses are held in hardware). Transit LSPs are identified by the use of swap in the static router entry, but this time in MPLS "label table" mpls.0.

```
admin@P9# show route table mpls.0 protocol static
1023               *[Static/5]  00:01:57
                   > to 10.0.79.1 via so-0/0/1. swap 1104
```

The link to P7 is so-0/0/1, as expected. The configuration on the P7, the second transit router, is very similar.

```
set protocols mpls interface so-0/0/1 label-map 1104 next-hop 10.0.17.1;
set protocols mpls interface so-0/0/1 label-map 1104 swap 1253;
```

If we wanted to configure PHP, this is the router where we would enable it. The statement swap 3 is the "magic word" that enables PHP. MPLS label value 3 says to the local router, "Don't really push a 3 on the packet, but instead pop the label and route the packet inside." The use of the label at least makes it easier to remember that the end of the LSP is really on PE1.

The Egress Router

The configuration on the egress router, PE1, is essentially the opposite of that on the ingress router but more similar to that on a transit router.

```
set protocols mpls interface so-0/0/2 label-map 1253 next-hop 10.0.12.0/24;
set protocols mpls interface so-0/0/2 label-map 1253 pop;
admin@PE1# set protocols mpls interface so-0/0/2 label-map 1253 next-hop 10.10.12.0/24;
admin@PE1# set protocols mpls interface so-0/0/2 label-map 1253 pop;
```

There is no need to tell the router what label value to pop: if it got this far, the label value is 1253. Note that the next hop is the IP address of LAN2, which is the entire point of the exercise. When PHP is used, there is no need for a label map for that LSP on the egress router. When PHP is not used, the egress LSPs are identified by the use of pop in the static router entry in mpls.0.

```
admin@PE1# show route table mpls.0 protocol static
1253               *[Static/5]  00:02:17
                   > to 10.10.12.0/24 via ge-0/0/3. pop
```

Static LSPs are fine, but offer no protection at all against link failure. And consider how many interfaces, labels, and other information have to be maintained and entered by hand. In MPLS classes, most instructors make students suffer through a complex static LSP configuration (some of which *never* work correctly) before allowing the use of RSVP-TE and LDP to "automatically" set up LSPs anywhere or everywhere. It is a lesson that is not soon forgotten. (In fact, dynamic LSP configuration using RVSP-TE is so simple that it is not even used as an example in this chapter.)

Traceroute and LSPs

How do we know that our static LSP is up and running properly? A `ping` that works proves nothing about the LSP because it could have been routed, not switched. Even one that fails proves nothing except the fact that something is broken.

But `traceroute` is the perfect tool to see if the LSP is up and running correctly. The following is what it looked like before we configured the LSP.

```
bsdclient# traceroute bsdserver
traceroute to bsdserver (10.10.12.77), 64 hops max, 44 byte packets
 1  10.10.11.1 (10.10.11.1)  0.363 ms  0.306 ms  0.345 ms
 2  10.0.50.1 (10.1.36.2) 0.329 ms  0.342 ms  0.346 ms
 3  10.0.45.1 (10.0.45.1) 0.330 ms  0.341 ms  0.346 ms
 4  10.0.24.1 (10.0.24.1) 0.332 ms  0.343 ms  0.345 ms
 5  10.0.12.1 (10.0.12.1) 0.329 ms  0.342 ms  0.347 ms
 6  10.0.16.2 (10.0.16.2) 0.330 ms  0.341 ms  0.346 ms
 7  10.10.12.77 (10.10.12.77)  0.331 ms  0.343 ms  0.347 ms
bsdclient#
```

Let's look at it now, after the LSP.

```
bsdclient# traceroute bsdserver
traceroute to bsdserver (10.10.12.77), 64 hops max, 44 byte packets
 1  10.10.11.1 (10.10.11.1)  0.363 ms  0.306 ms  0.345 ms
 2  10.0.59.1 (10.0.59.1)  0.329 ms  0.342 ms  0.346 ms
 3  10.0.16.2 (10.0.16.2) 0.330 ms  0.343 ms  0.0347 ms
 4  10.10.12.77 (10.10.12.77)  0.331 ms  0.343 ms  0.347 ms
bsdclient#
```

Only four routers have "routed" the packet. On the backbone, the packet is switched based on the MPLS tables, and so forms one router hop. But at least we can see that the packets are sent toward P9 (`10.0.59.1`) and not P4 (`10.0.50.1`).

The details of the path of MPLS LSPs are not visible from the hosts. Why should they be? LSPs are tools for the service providers on our network. Only on the routers, running a special version of traceroute, can we reveal the hop-by-hop functioning of the LSP. When run on PE5 to trace the path to the link to CE6, traceroute "expands" the path and provides details—showing that the CE6 is still five routers away from CE0 (and that there are still six routers and seven hops between LAN1 and LAN2).

```
admin@PE5> traceroute 10.10.16.1
traceroute to 10.10.12.0 (10.10.12.0), 30 hops max, 40 byte packets
 1  10.10.12.1 (10.10.12.1)  0.851 ms  0.743 ms  0.716 ms
    MPLS Label=1023 CoS=0 TTL=1 S=1
 2  10.0.59.1 (10.0.59.1)  0.799 ms  0.753 ms  0.721 ms
    MPLS Label=1104 CoS=0 TTL=1 S=1
 3  10.0.79.1 (10.0.79.1)  0.832 ms  0.769 ms  0.735 ms
    MPLS Label=1253 CoS=0 TTL=1 S=1
 4  10.0.17.1 (10.0.17.1)  0.854 ms  0.767 ms  0.734 ms
 5  10.0.16.1 (10.0.16.1)  0.629 ms !N  0.613 ms !N  0.582 ms !N
admin@PE5>
```

Just to show that the LSP we set up is unidirectional, watch what happens when we run `traceroute` in *reverse* from `bsdserver` on LAN2 to `bsdclient` on LAN1.

```
bsdserver# traceroute bsdclient
traceroute to bsdclient (10.10.11.177), 64 hops max, 44 byte packets
 1  10.10.12.1 (10.10.12.1)  0.361 ms  0.304 ms  0.343 ms
 2  10.0.16.1 (10.1.16.1) 0.331 ms  0.344 ms  0.347 ms
 3  10.0.12.2 (10.0.12.2) 0.329 ms  0.340 ms  0.345 ms
 4  10.0.24.2 (10.0.24.2) 0.333 ms  0.344 ms  0.346 ms
 5  10.0.45.2 (10.0.45.2) 0.329 ms  0.342 ms  0.347 ms
 6  10.0.50.2 (10.0.50.2) 0.330 ms  0.341 ms  0.346 ms
 7  10.10.11.177 (10.10.11.177)  0.331 ms  0.343 ms  0.347 ms
bsdclient#
```

Packets flow through backbone routers P2 and P4, as they did before the MPLS LSP was set up! The "old" route is used, showing that MPLS is the basis for traffic engineering on a router network.

QUESTIONS FOR READERS

Figure 17.9 shows some of the concepts discussed in this chapter and can be used to help you answer the following questions.

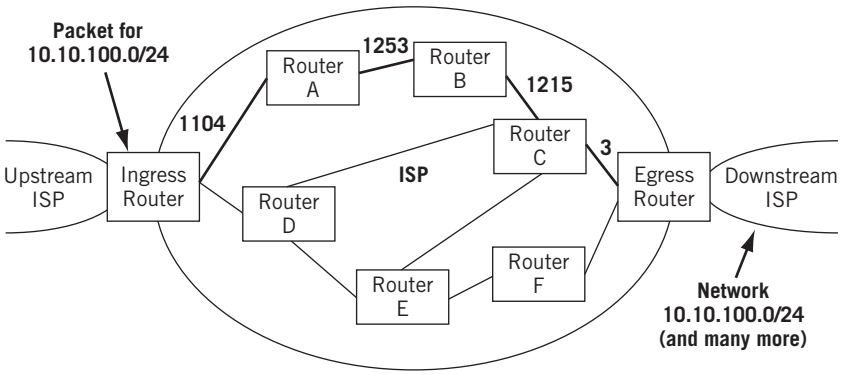


FIGURE 17.9

An MPLS LSP from ingress to ingress router, showing label value to path. The LSP runs along the heavy lines through the routers designated. The label values used on each link are also shown.

1. Does the LSP in Figure 17.9 use the shortest path in terms of number of routers from ingress to egress?
2. What does *traffic engineering* mean as the term applies to MPLS?
3. Is there an LSP set up on the reverse path from egress to ingress router?
4. Which label is used on the LSP between routers A and B? Is this label added to another, or swapped?
5. Is PHP used on the LSP? How can you tell?