

1

Introduction

“It is a mistake to look too far ahead. Only one link in the chain of destiny can be handled at a time.”

—Winston Churchill

Introduction

In this chapter, we look at the origins of Multi-Protocol Label Switching (MPLS) and introduce some of its basic concepts, including the separation of the control and forwarding planes of MPLS, the Forward Equivalence Class, and the MPLS label. After this introduction, we then introduce and discuss some of the new applications of MPLS networks such as traffic engineering and virtual private networks.

After an introduction to MPLS, we explain the basic premise behind why MPLS-enabled networks need to be managed to provide scalable, usable, and most importantly *profitable* MPLS networks. Given this motivation, we introduce how MPLS networks

can be managed effectively using both standards-based and nonstandard tools, many of which are described in this book. This discussion serves as an introduction to the remainder of the book.

It is not our goal for this discussion to be an in-depth introduction to MPLS. We assume you have a good level of understanding of MPLS already and that the introduction given in this chapter can be used as a refresher. Advanced readers may skim the beginning of the chapter, but we recommend at least glancing at the latter half of the chapter. If you are in need of a more in-depth introduction to MPLS and SNMP, consult the references given in the Further Reading section at the end of the chapter as well as those related to MPLS and SNMP in the Bibliography at the end of the book.

1.1 A Brief Introduction to MPLS

In the past, routing devices were designed with the control and forwarding components commingled, which led to many shortcomings including low performance and scalability issues. In particular, routing lookups, especially those involving so-called longest-prefix match lookups, were quite complex and expensive in nature—in fact, quite a deal more complex than any layer-2 switching or bridging operation. Further complicating this process was the fact that many routers were required to forward packets from many different routing protocols. By accepting packets from different protocols, the positions of fields in packet headers could potentially be different for nearly every packet received, potentially further degrading forwarding performance. In contrast, nonrouting devices such as layer-2 bridges and switches were able to forward traffic at relatively high speeds because they based their forwarding decisions not on variable-length packet headers and network addresses of varying lengths, but on a short, fixed-length field. For example, all ATM cells have a fixed length and well-defined format. Devices switching ATM cells only need to examine a short identifier and can immediately forward the cell based on this simple piece of information. There is no question as to the position of the forwarding information in a cell. However, layer-2 devices suffered from the lack of routing information, which ultimately limited their scope and effectiveness. Let us now examine the control and forwarding planes in more detail, and then investigate how they can form the basis of an efficient and scalable MPLS label switching router (LSR).

The control component of a router is responsible for the exchange of routing information between other network nodes. It is this information that is used to form the router's routing database. This database paints a picture of the network from which a router can discern what it considers to be the most optimal path to any given destination in the network. Once stabilized, this database of best paths can be used to program the router's forwarding table. In contrast, the forwarding

function of a router focuses exclusively on the actual decision of moving packets between ports on a network node. Each packet contains a header with source, destination, and other information. When a node receives a packet on a port, it needs to decide which port (or ports) it needs to forward that packet to. The forwarding process is quite mechanical by nature. When a node receives a packet, the forwarding component in that node will first examine the destination address contained in the incoming packet as well as perhaps other fields in the header. This information is then compared with entries in its forwarding database. It is this simple process that allows the forwarding component to make quick and simple decisions as to where the packet needs to be forwarded.

In some devices, the forwarding component is tightly coupled with the routing component. This approach sometimes results in limited portability of that technology to other types of forwarding planes. It also sometimes results in difficulties in extending the protocol with additional functions. MPLS is built on both the premise of a clean separation of the control and forwarding functions to take advantage of their individual advantages, as well as using them together in concert to provide additional advantages not possible with other technologies. The control and routing functions of MPLS are based on the Internet Protocol suite of protocols, which includes IP, RSVP, BGP, OSPF, and so on. The basic device in an MPLS-enabled network is the LSR. This device implements both the MPLS control and forwarding planes. The control function of an MPLS label switching router is responsible for distributing routing information to other LSRs, as well as the information required to convert this information into forwarding tables that can then be used by the forwarding function. The MPLS forwarding function is based on the use of a short, fixed-length label. This concept comes from the use of the same concept in layer-2 technologies such as ATM and Frame Relay, which base forwarding actions on a short, fixed-length identifier.

1.1.1 Forward Equivalency Classes

The forwarding function of a router is responsible for forwarding traffic toward its ultimate destination. The information in the forwarding table is programmed based on information from the control plane. If a packet is not delivered via a local interface directly to the destination, the router must forward the packet toward the ultimate destination using a port that will steer that traffic on a path considered most optimal by the routing function. For this reason, a router must forward traffic toward its destination via a next-hop router. This next-hop router may be the next-hop along the most optimal path for more than one destination subnetwork, so many packets with different network layer headers may be forwarded to the same next-hop router via the same output port. The packets traversing that router can then be organized into sets based on equivalent next-hop network nodes. We call

such a set a *Forward Equivalency Class* (FEC). Thus, any packet that is forwarded to a particular next-hop is considered part of the FEC, and can thus be forwarded to the same next-hop. One important feature of the FEC is the granularity of the classification of traffic it can encompass. Since the FEC is based on a routing next-hop, it can include different classifications of packets. For example, since the routing information for a particular next-hop classification can be based on a destination prefix, it might include every packet traveling toward that destination. In this way, the granularity of packets classified by that FEC is quite coarse. However, if the routing database has programmed some next-hops for some traffic based on an application layer, for example, the traffic granularity might be much finer.

1.1.2 The MPLS Shim Header

MPLS packets are encapsulated using an MPLS shim header. The header has this name because it defines an additional header that is placed—or shimmed—between existing layer-2 and layer-3 headers. Curiously, the verb comes from the noun, not the other way around. Therefore, the shim header is so called because it is a small object that is inserted—shimmed—in between the existing layer-2 and layer-3 headers. Figure 1.1 shows the MPLS shim header format. The shim header comprises a sequence of one or more label stack entries. The entries in the sequence can be viewed together as a conceptual stack. A label stack entry comprises several components: label, EXP bits, the bottom of the stack bit, and TTL. The first element is the MPLS label. The label is a fixed-length 20-bit quantity that represents the label used to switch a packet. This label has local significance on a given interface between two neighboring LSRs only. That is, a label taken out of the context of a specific interface between two LSRs may or may not be found to be useful, or may be assigned to a different segment of an LSP. The second portion of the header is 3 bits called the experimental (EXP) bits. These bits are reserved for experimental use, such as for the purposes of classifying LSPs using Differentiated Services code points. The next element of the shim header is a single bit used to indicate the “bottom of the stack.” This bit is set to 1 for the last entry in the label stack (i.e., for the bottom of the stack) and 0 for all other label stack entries. The fourth and final element in the stack is an 8-bit quantity called the time-to-live (TTL) field. The format of a label stack entry is detailed in Figure 1.2.

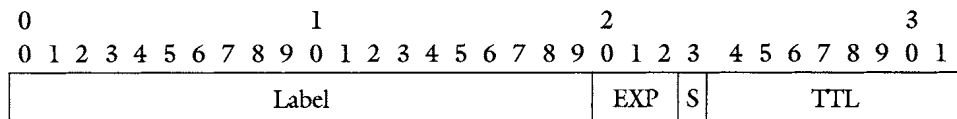


Figure 1.1 MPLS shim header format.

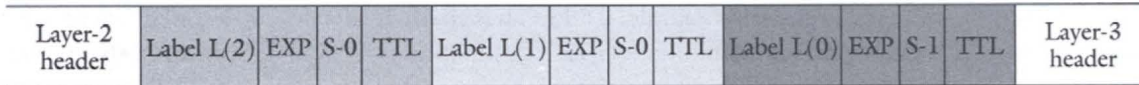


Figure 1.2 The MPLS label stack as it appears within the MPLS shim header.

The Label Stack

MPLS packets may contain more than one label. Depending on the application, it may be desirable to nest label switched paths (LSPs). For example, some TE and VPN operations find it useful to nest LSPs. When labels are nested, they are represented in the MPLS shim header as a stack structure, that is, a LIFO queue. The label stack is represented as a sequence of “label stack entries” in this stack. The topmost label appears closest to the layer-2 header, and the bottommost closest to the layer-3 header. Figure 1.2 demonstrates the label stack as a sequence of label stack entries. Each label stack entry is represented by 4 octets or 32 bits of data. Only the topmost label stack entry is used for any single lookup in the MPLS Label Forwarding Information Base (LFIB).

1.1.3 MPLS Label Switching

The MPLS forwarding plane is responsible for forwarding traffic based on an MPLS label. An MPLS label is a short, fixed-length 20-bit value (see Figure 1.1). The MPLS label has no structure. The MPLS label only has local significance between any two LSRs; therefore, the same label can be reused simultaneously within an MPLS-enabled network. In order for an MPLS LSR to be able to switch an MPLS packet, the label used in that packet’s header must represent an entry in the MPLS LFIB of that LSR. The LFIB is essentially the label-to-label switching database used to program the LSR’s forwarding plane. Once a packet is received, its label will be used by the forwarding plane to make a decision on where to forward the packet. At the edges of an MPLS-enabled network, label switching routers will map IP packets into FECs based on information provided by the MPLS control plane. Once classified into a FEC, the forwarding plane will be able to encapsulate any packet it receives that matches that FEC using the next-hop MPLS label assigned to that FEC.

Although assigned to a particular packet, the MPLS label does not necessarily encode the packet’s network layer address, just its next-hop that will allow the packet to be forwarded to its destination, because many packets that are in the same FEC will be assigned the same label. This means that the next-hop choice may span multiple packets to many destinations. Thus, an MPLS label really encodes a

FEC identifier. For example, if a FEC has classified all packets destined for the same next-hop based on multiple layer-3 destination network prefixes, all of the packets matching that FEC will be assigned the same label (and next-hop). Once a packet is assigned a label, it will be switched based on this label until it reaches its ultimate destination. At that point, the MPLS header is removed and the packet forwarded using its original encapsulation.

When an MPLS packet is received, the LSR attempts to find a matching forwarding entry in its LFIB based on the packet's label and the interface on which the packet was received.

There are three operations that may be executed on the label stack when an MPLS packet is received and an entry matching this label is found in the LFIB: *push*, *pop*, or *swap*. All operations are executed on the top entry of the stack. When the topmost label is "popped" from the label stack, its label stack entry is completely removed from the MPLS shim header. When a label is "pushed" onto the stack, it moves all of the existing labels down by one relative index in the stack and inserts itself at the top of the stack. When a swap operation is executed, the topmost label entry is replaced with a different label, but the size of the stack remains the same. The "S" bit is set to indicate the last or bottommost entry in the label stack. All other entries in the label stack must set the "S" bit to 0.

The example shown in Table 1.1 demonstrates what an MPLS LFIB might look like. In the example, labels that are received on this LSR's MPLS interface "MPLSEth1/2" are switched to various other interfaces based on the incoming label. For example, when a packet containing label "1000" is received on interface "MPLSEth1/2," it is swapped for label "1050" and is forwarded on interface "MPLSEth1/3" to next-hop address 10.20.0.1. Note that this happens in all but the second-to-last row. The outgoing label in this case is noted as "pop." This refers to the removal of the MPLS shim header from the packet. Packets that have their MPLS headers stripped or "popped" are then forwarded on using their layer-3 encapsulation.

Table 1.1 Simplified Label Forwarding Information Base.

Incoming interface	Incoming label	Outgoing label	Next-hop	Outgoing interface
MPLSEth1/2	1000	1050	10.20.0.1	MPLSEth1/3
MPLSEth1/2	1002	1070	10.30.0.1	MPLSEth1/6
MPLSEth1/2	1006	"pop"	—	—
MPLSEth1/2	1005	1080	10.40.0.1	MPLSEth1/7

The MPLS Domain

An MPLS domain is composed of one or more MPLS label switching routers. A label switching router is any router or switch that supports the forwarding of MPLS-encapsulated packets based solely on the incoming interface and the information in the shim header. An LSR that sits at the edges of an MPLS domain and forwards traffic into and out of the MPLS domain is called a label edge router (LER). An LER maintains at least one interface into and out of the MPLS domain and acts as the point where the MPLS shim header is first *imposed* onto the incoming packet, and where the header is ultimately stripped and the packet forwarded using its original layer-3 encapsulation.¹ In effect, the LER must connect between the incoming technology and MPLS or vice versa. This process, in effect, tunnels the incoming technology through the MPLS network by encapsulating it within the MPLS packets.

Figure 1.3 shows a simple MPLS domain as well as the basic components of an MPLS-enabled network. The figure shows how MPLS LERs connect to external IP networks that may or may not contain customer sites. LERs are interconnected with other LERs within an MPLS-enabled domain. Other MPLS LSRs are interconnected in various ways within the MPLS domain.

The Label Switched Path

The path taken through the MPLS domain by a packet is referred to as a *label switched path* (LSP). The path taken may not be understood or completely stored by any one LSR within the MPLS domain, although in some cases it is. For example, traffic engineering allows the complete path to be stored at all LSRs along the path. This is because the labels swapped at each LSR have only local significance with regard to any two adjacently connected LSRs. Each LSR simply makes a local forwarding decision based on the incoming label of a packet, and switches the packet to a known outgoing label on a different interface. We should note that once the LFIBs have been established on all LSRs along the path of an LSP, the LSP is uniquely associated with the label and interface it is associated with, and therefore it is uniquely associated with a FEC.

An example of a label switched path is demonstrated in Figure 1.4. IP traffic to a destination reachable via the second LSR from the left is bound to a FEC at the leftmost LSR. All traffic entering the leftmost LSR will be classified using this FEC and will subsequently have MPLS shim headers imposed with a specific label

¹ From this point on, we will assume that the layer-3 payload is always IP. Other protocols are equally supported and handled by routing, switching, or forwarding engines specific to their characteristics. However, to avoid confusion in the text we will limit our view to the most common payload, which is IP.

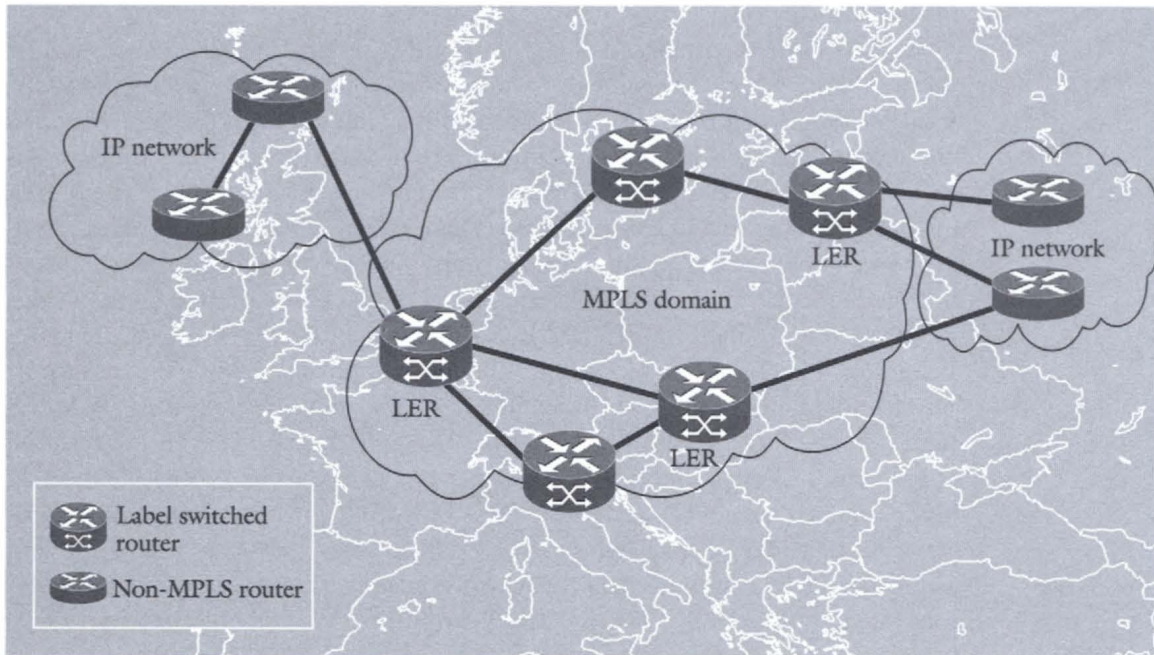


Figure 1.3 Components of an MPLS network.

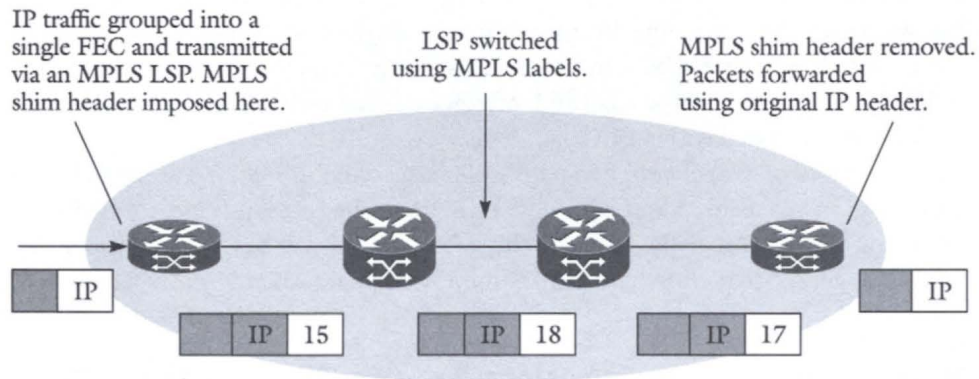


Figure 1.4 Example of an MPLS label switched path.

associated with this FEC—in this case 15. The MPLS-encapsulated packet will leave the leftmost LSR and will have its label swapped with the one indicated on the link as it traverses the LSP. When it arrives at the rightmost LSR, the shim header is removed and the packet forwarded out the rightmost interface using its original encapsulation.

1.2 Applications of MPLS

Currently, the two most important applications of MPLS are traffic engineering (TE) and virtual private networks (VPN). However, other new applications are taking shape such as Differentiated Services-aware traffic engineering that will enable voice-over-IP applications over MPLS, as well as virtual circuit emulation and virtual private LAN services over MPLS networks that will allow existing MPLS networks to be leveraged to offer additional emulated services. Although some of the applications of MPLS, such as traffic engineering and virtual private networks, technically can and are, in fact, currently being implemented and deployed using existing non-MPLS-based protocols, MPLS makes these applications simpler and more scalable. At the heart of the reasons for why MPLS is able to achieve these goals is because it takes advantage of the separation of the routing and forwarding functions, and because of its integrated signaling mechanisms. This has the advantage of reducing or eliminating many of the limitations of traditional routing and provisioning. For example, in the use of virtual private networks, MPLS simplifies the act of configuring a VPN by only requiring that the operator configure the edge devices connecting the customer edge networks into the VPN. MPLS signaling takes care of the actual connection to other pieces of the VPN. MPLS further improves the scalability by obviating the need for state information about the VPN to be stored anywhere within the core of the network. An example of an MPLS network that supports VPN is depicted in Figure 1.5. In this example two VPNs are supported: VPN A and VPN B. In order to support each VPN, the provider edge (PE) devices that connect the VPN sites must be configured. The core of the network is composed of provider core LSRs, or “P” routers. For example, P1 denotes a core “P” router in the figure. The core “P” routers do not have their configurations modified to support new sites of VPNs.

Another example of an important application of MPLS is in traffic engineering, where through the use of MPLS, it is possible to specify explicit routes during the process of setting up a path such that some specific data may be routed around network hot spots. Current technologies use routing protocols that tend to converge on a single, least-cost path to each possible (aggregate) destination. This occurs

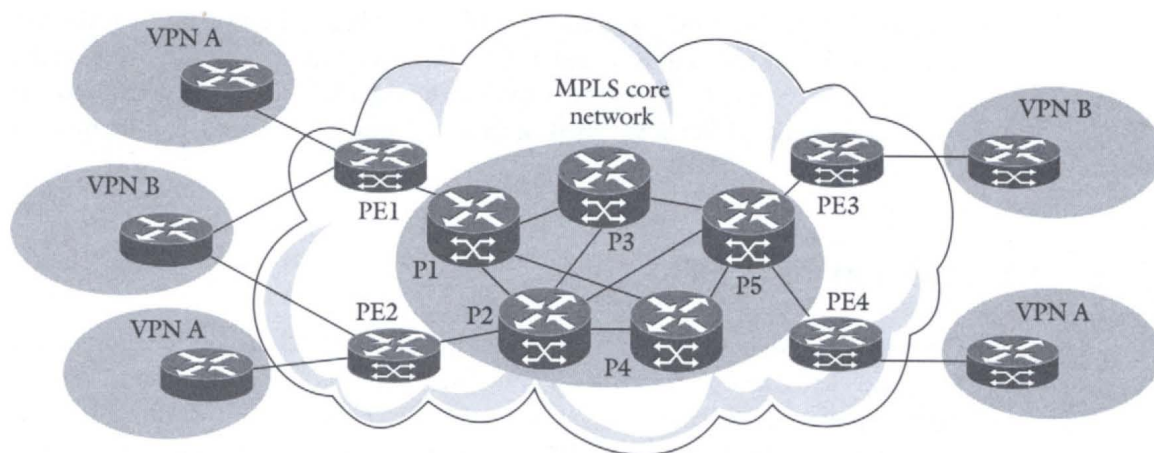


Figure 1.5 Example of an MPLS VPN.

even if there are multiple least-cost paths to the same destination. There are several problems with this approach. First, in many cases, parallel equal-cost paths exist to the same destinations, but all but one is preferred by the routing protocol. Second, since protocols generally prefer a single path that is considered most optimal, the routing protocols will direct all of the traffic destined to that destination onto that path. This often results in network hot spots at points in the network where many paths cross a single node.

It is possible to overcome these shortcomings with MPLS traffic engineering. MPLS TE allows an operator to specify an explicit route to direct some fraction of traffic through other parts of the network that are not selected by the routing process. These alternate paths may or may not be parallel least-cost paths. The important point is that the operator has the ability to override the routing protocol and choose which path certain flows of traffic take. Furthermore, TE allows an operator to create alternate backup paths, which bypass network trouble spots (i.e., disabled nodes or links). Given this mechanism, it is also straightforward to establish MPLS TE tunnels that transport packets that would not otherwise be correctly routed across a backbone network. For example, this is sometimes necessary in order to support virtual private networks across a backbone network between VPN end points, thereby making address translation and more costly tunneling approaches unnecessary.

An example of MPLS traffic engineering is depicted in Figure 1.6. Assume that each link carries an equal cost that is given to the routing protocol. Notice that given this assumption, two equal-cost paths that traverse the same number of network nodes exist. The thick dotted line represents the path through the network

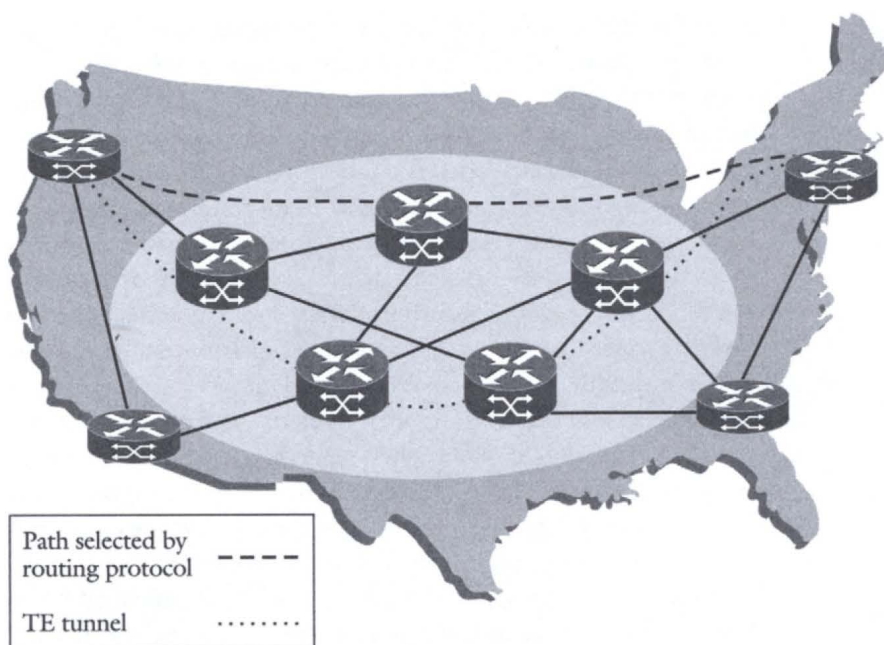


Figure 1.6 Example of a MPLS traffic-engineered tunnel.

that the routing protocol has chosen as most optimal. The thin dotted line represents an MPLS traffic-engineered tunnel that has been configured to override the path chosen by the routing protocol. This allows some of the traffic that would have taken the default path to be steered across the alternate path. In this example, the TE tunnel has been configured to use an alternative, unused path through the network in an effort to better utilize network resources.

1.3 Key Aspects of MPLS Network Management

Networks need to be managed for several reasons. First, from an entirely practical perspective, devices need to be monitored to ensure that they are functioning properly. Devices may also alert the operator to fault conditions, but if no corrective action is taken by the operator, then the device may continue to malfunction. For example, if a router's routing table has grown to a size that will soon exceed its available memory, it may be beneficial for the device to inform the operator of this condition. Services that are offered by a network also need to be managed, particularly when they are provisioned. In these cases, devices are contacted and configured. Managed services also require monitoring and maintenance. For example, if a

service provider offers a virtual private network service to end users, it may be necessary to monitor the health and performance of the network paths that carry that customer's traffic to ensure that they are getting the network services that they paid for. In fact, this monitoring arrangement is sometimes a contractual necessity.

In all of these scenarios, it is either extremely difficult or nearly impossible for operators of medium to large networks to monitor every device in their networks by hand; instead, most prefer to do this in an automated manner. Some accomplish their management using a centralized approach, as is demonstrated by the sophisticated operations center shown in Figure 1.7. However, others may choose to have several smaller operations centers that are distributed. In either case, it is extremely time-consuming, and hence costly, for an operator to manually connect to each device's console in order to monitor its status, isolate faults, or configure the device. This becomes more obvious when you consider those provider networks where the network devices are located over a wide range of geographic areas. In this case, it becomes even more costly to travel to a remote location or hire additional staff to be on site where those additional devices are located.

Second, to make a sound business case for deploying MPLS, it must be made fully manageable so that the operational aspects of the network can scale up to numbers of devices, services, and customers that will make the network profitable. For example, the money spent debugging a problem by sending an operator into the field or by having the same operator go from router/switch to router/switch scratching their head might be better spent in building an automated system that can listen for alarms (see Section 2.6.6) that the router/switch can emit when in

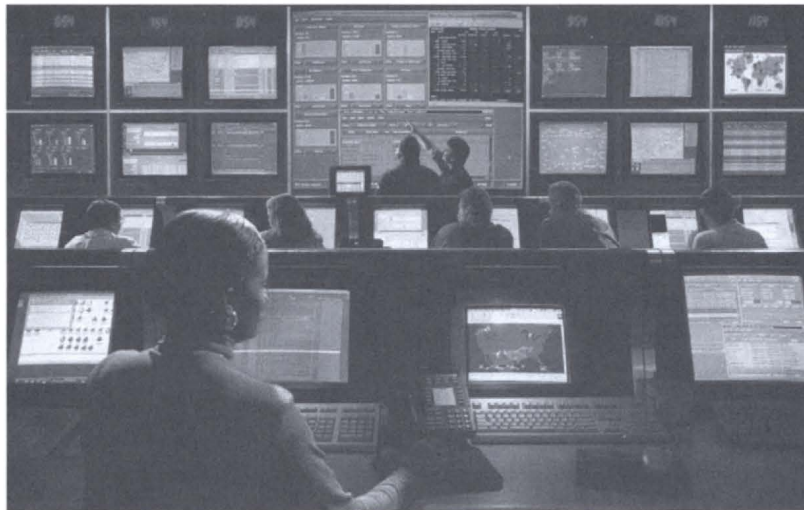


Figure 1.7 A network management operations center.

distress. These alarms can then be used to pinpoint and isolate the scope of the problem. Once isolated, a management system can even take automated actions to correct the situation or simply alert an operator. An automated system can even be smart enough to not bother an operator if it deems a problem insignificant. Furthermore, management of the MPLS network becomes paramount when placed within the context of service level agreements and MPLS VPN services (see Chapter 11). When service level agreements are made between customers and providers, the service provider will not earn any money from that customer unless the services provided meet the agreement. The monitoring of the agreed-upon terms such as bandwidth, latency, delay, or service availability can be best accomplished using a network management system.

1.3.1 Origins of Network Management for MPLS

Once MPLS began to become mature and operational experience began to be gained by service providers deploying the technology, it was clear that MPLS was not very manageable given the lack of standard tools and management interfaces available at the time. In particular, the majority of MPLS vendors including Juniper and Cisco had only provided proprietary command-line interface extensions for the configuration and monitoring of MPLS features. When MPLS deployments were in early stages, it was acceptable for these and other vendors to provide minimal management capabilities for the MPLS features since operators were largely interested in simply having the protocol function up to specifications. However, as deployments became more mature and providers were more comfortable with the notion of using this protocol, it was clear that management of the protocol and its many features was now a priority. Furthermore, in heterogeneous networks where devices from multiple vendors had to coexist, an even larger problem existed. Since vendors had only deployed proprietary command-line interfaces, providers deploying devices from more than one vendor had to contend with more than one management interface for MPLS. This approach is expensive because it requires duplication of effort to manage the configuration and monitoring of the same features. The duplication of resources often ultimately translates into lost revenues for service providers. It was these requirements that began the push for standard interfaces for MPLS. In particular, the work on the IETF MIBs began in earnest during this time.

1.3.2 Configuration

One sore point for many operators is how to configure each one of the potentially hundreds of devices in their network. Further complicating the picture of configuration is the fact that many, if not most, provider networks are not comprised of

devices made by a single vendor. This results in the service provider having to learn at least one different configuration language for each vendor from which it purchases equipment. Even further compounding this situation is that, through the magic of mergers and acquisitions, many vendors actually supply devices that have different configuration languages depending on which product line of theirs you choose to deploy.

It should be obvious from this description of the problems inherent in configuring a network of devices that it is a difficult situation at best. What would alleviate this situation would be the use of a common language and associated interfaces that can be used for the configuration of devices. There are many such languages available, yet no single one is used ubiquitously. Perhaps the closest contenders are SNMP—that is, SNMPv1 (RFCs 1155, 1157, and 1213), SNMPv2c (RFCs 1901–1906), and SNMPv3 (RFCs 2571–2575)—CORBA, and XML. Unfortunately, today the clear winner, at least for configuration, is the proprietary command-line interface (CLI), although SNMP is generally regarded as the best option for monitoring. The difficulty with a proprietary CLI is that it is generally accessible only via telnet or hardwired connections and generally has no standards-based schema. This results in every vendor implementation having a different management interface, which is clearly not something that excites a provider deploying a multivendor network. Although the CLI represents a majority of management interfaces, at least in the configuration area, the tide is turning toward standardized interfaces as networks grow ever more complex. These interfaces are commonly used for monitoring, and in many cases for provisioning as well. We will delve into the details of these various standard mechanisms for configuration in the pages to come.

1.3.3 Service Level Agreements

Typically, when a user signs up for access service (e.g., DSL, cable modem, dial-up), the service provider only agrees to provide that user with access to their network, and sometimes eventual access to the Internet. This agreement typically only specifies a minimum amount of bandwidth and provides no specifics about the average delay between access points and any other point in the network, or generally any other guarantees of service. Furthermore, there is typically no minimum response time during which outages in the network will be corrected by the service provider. This generally means that the user of a service is out of luck if their service does not function as advertised.

Some operators take their level of service a step further. These operators choose to monitor and maintain what some refer to as the “user experience.” Although many operators strive to have networks simply function (i.e., route and switch a lot of traffic), others wish to ensure that their network is performing at levels

acceptable to its customers. For example, this can mean that if user access to the Internet is unacceptably slow, the service provider will take some action to correct the situation—sometimes automatically. This approach is in direct contrast to other providers who would be content with end users just having access to the Internet at any speed.

The notion of service assurance and verification can be taken a step further beyond a provider assuring that they will monitor the health of user services. Frequently, end users and service providers will enter a formal contract called a *service level guarantee* or *agreement* (SLA). This agreement is an official agreement or contract between the service provider and a customer that specifies that the provider will sell a certain service to an end user for a certain price. If this service is provided as agreed upon, the end user must pay a certain fee for the service. However, if the service is not provided, typical recourses for the user are a reduction or refund of the fee they pay for the service during that period. Often the amount of additional work that a provider must perform to ensure that a service is functioning according to the service level agreement is significant. This elevated cost is precisely why SLAs are typically only signed between service providers and higher-paying customers such as large corporations or other service providers.

For example, in the United States, the service provider market is largely focused on selling bandwidth. This bandwidth is sometimes sold with guarantees of quality such as minimum delay and jitter. In other parts of the world, service providers concentrate instead on selling VPN services where site-to-site access quality is most important. All of these deployments typically contain SLA agreements with guarantees on the components of the service that the customers find most important, as well as the things a provider is willing to assure.

Given the motivation and elevated revenues from SLA agreements, providers are motivated to offer these premium services. However, these services do not come without additional effort on their part to verify the service quality and take corrective action when it does not meet the specified quality. In this regard, manual verification of SLAs is highly undesirable from a provider's perspective. This is simply because of its repetitive and frequent nature, especially when performed on a large scale. SLA agreements may also require that the operator take corrective action within some short period of time after a fault is detected. It is for these reasons that SLA monitoring and verification can be cumbersome or impossible if done manually, and therefore is a driver for the task to be performed by a fully or semi-automated network management system. In order to realize a management system that can verify SLAs in an automated fashion, network management functions must be integrated into devices that must be monitored. In particular, common management interfaces allow a provider to effectively monitor the data points of a service. This is especially important for heterogeneous networks and is also important in

cases where customers insist on having independent third parties verify the SLA, since these companies often prefer not to build SLA verification software that is customized to a particular provider's network. Instead, they prefer to build software that is able to talk to a large set of devices in order to service many different service provider networks.

1.3.4 Service Level Agreement Verification

One often-overlooked aspect of service level agreement contracts is called service level agreement verification. The agreement of services between the end user and provider can be verified in several ways. The simplest form might be to issue Internet Protocol (IP) “pings” that emanate from the customer access points to other points in their networks or to locations within the Internet. This simulates user traffic traveling along the data path that all traffic takes through the network. If this traffic takes too long to traverse the network—or worse, is not getting to certain points within or external to the service provider's network—then the user experience suffers. Monitoring of the user experience might also be as sophisticated as monitoring the performance of many key network devices, collecting this information at a central location, and then making dynamic adjustments to the network using this information.

More sophisticated SLA verification is typically accomplished using network management tools that are specifically designed for the task. These tools include remote monitoring (RMON) or simply monitoring various counters on the network devices. Figure 1.8 illustrates how SLA monitoring and verification might be accomplished within an MPLS VPN deployment. A network management system (NMS) is positioned at key points, monitors certain traffic and quality of service (QoS) statistics, and reports them to the operator and customer. SLA verification can be done by the service provider, the customer, or by an unbiased third party. Use of standard network management interfaces to expose variables within the often-diverse population of network devices present in service provider networks is critical, especially when a third party is contracted to do the verification. The reason for this is simple: interoperability. SLA verification becomes quite cumbersome and costly if the party performing the verification is required to customize the verification suite for every device in a network. This is important if a third-party SLA verification company either sells software/hardware to service providers or performs the SLA verification service directly.

1.3.5 Fault Isolation

Fault isolation and detection are simply a means by which operators can detect, isolate, and report on defects discovered within their networks. The operator can use

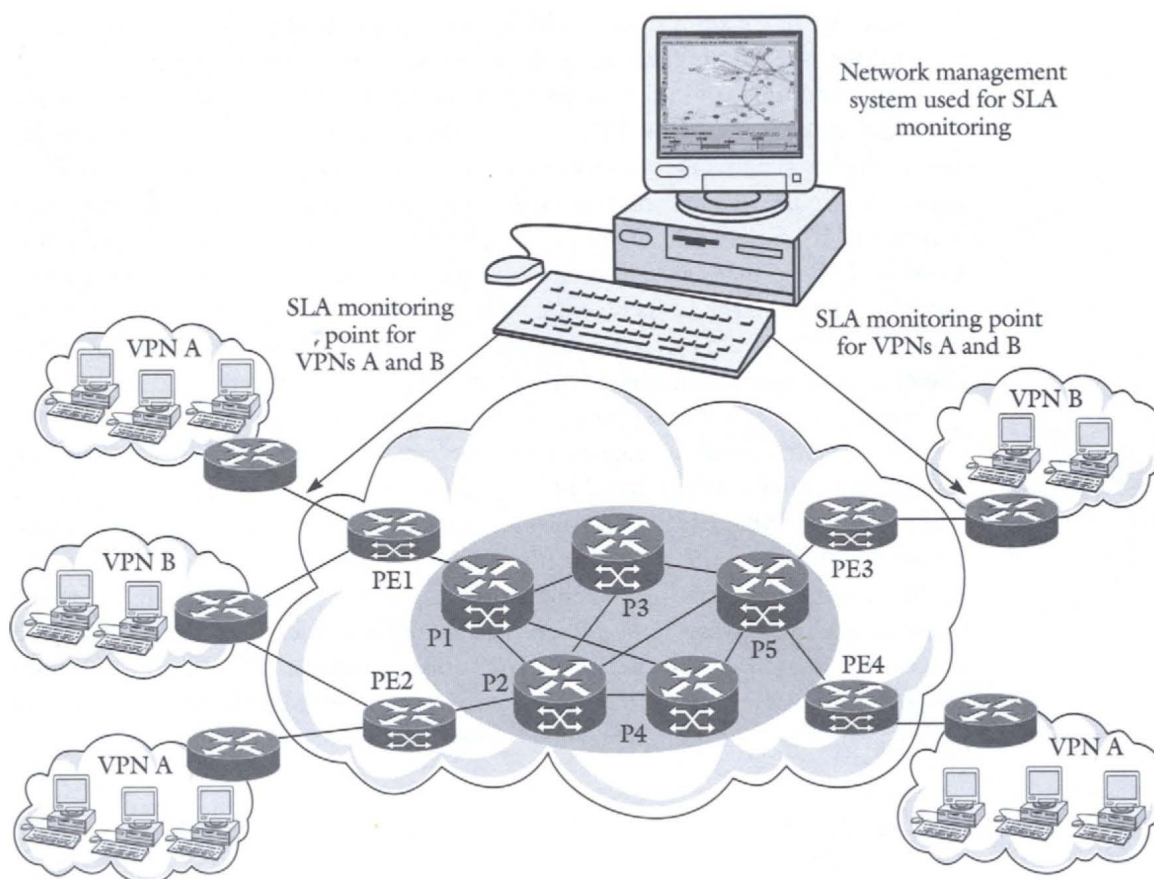


Figure 1.8 Example of a network of VPNs where SLAs are monitored by gathering information at certain key points within the network.

the information to repair the defect(s) found manually or automatically. When a device detects a problem, it will emit one or more messages as an alarm to alert the operator of the fault condition. These messages can be emitted under many conditions, including loss of service, device in distress (e.g., low on memory), or when the device has rebooted. Fault isolation is usually accomplished in modern networks in a three-part process that includes devices emitting asynchronous alarms, operators receiving those alarms, and then taking possible action because of those alarms. When a network device such as a router or switch discovers that an event of interest has occurred, it may issue an alarm. This alarm can be of the form of a system console message or an SNMP notification, which can be transmitted to the operators as an inform or notification. The reason for raising the alarms can include a configured threshold being exceeded, an internal fault condition such as

low memory, or a system reboot. Although other forms of alarms do exist, including audible buzzers or flashing notifications on the command terminal, SNMP notifications are used in the majority of deployments. Depending on the size and structure of the service provider's network, the operator may place one or more listening probes (i.e., workstations) around their network to listen and collect these messages. Figure 1.9 demonstrates such a configuration where an NMS is deployed within an MPLS network. One of its purposes is to listen for notifications emitted from the LSRs in that network. The figure shows one of the links in the sample network breaking and the LSRs on either side of that link emitting an SNMP notification. The NMS would catch this notification and possibly alert the operator to the situation or trigger an automated procedure for possible corrective action.

Sometimes, when the networks are large and/or multitiered, the operator will even have notifications aggregated and perhaps even summarized if processing power permits, and then relayed to a central alarm-processing center. This center will then decide whether or not to issue a trouble ticket for an alarm and dispatch personnel to address the situation. It should be obvious that the activities just described would be next to impossible to achieve if done manually in any practical network deployed today. It should be clear that in order for MPLS to be deployed successfully on a large scale, network nodes must be capable of issuing the necessary alarms (i.e., SNMP notifications) that are specific to not only MPLS functions, but also the other functions in the devices being deployed.

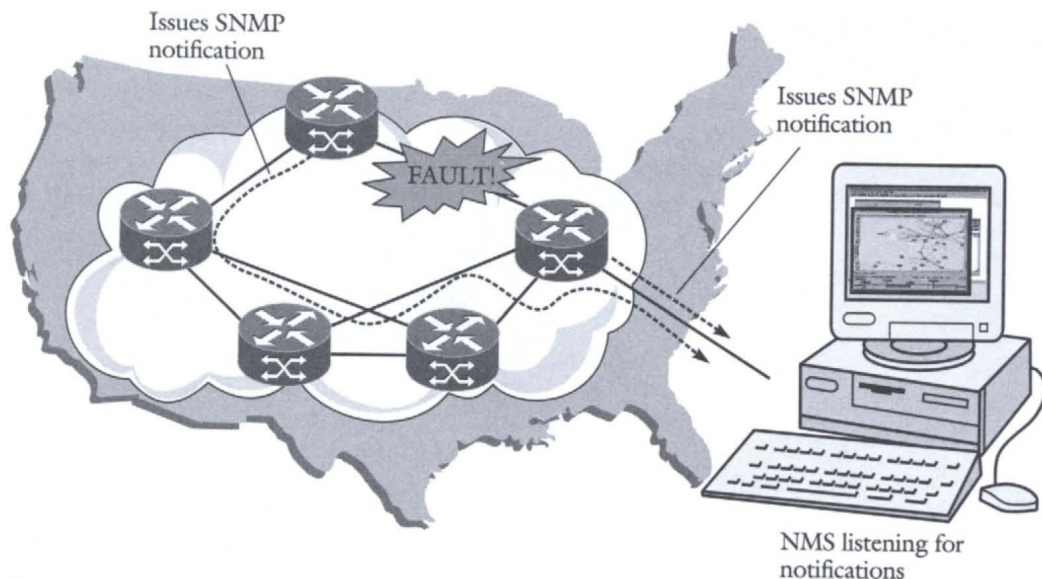


Figure 1.9 MPLS fault isolation using SNMP.

1.4 Management Information Base Modules for MPLS

The IETF, ITU, ATM Forum, and other standards bodies define documents called Management Information Base (MIB) modules that provide an external management interface for protocols and other features that are standardized within those organizations. Each MIB module can be thought of as a form of a data model used to manage the protocol or feature. The MIB module also defines the syntax, maximum access levels, and object interactions between those objects defined in that and other MIBs. The collection of MIB modules comprises the conceptual MIB that defines the entirety of MIB modules. We should also note that a MIB module is sometimes referred to as “a MIB” within certain contexts; thus care should be taken to discern when you mean a single MIB module or a collection of MIB modules that comprise a MIB.

The MPLS Traffic Engineering MIB module (MPLS-TE MIB) and the MPLS Label Distribution Protocol MIB (MPLS-LDP MIB) were the first MIBs proposed at the IETF in 1998. As standards-related work on these MIBs continued within the Working Group, and implementation and operational deployment by both device vendors and service providers continued, the MPLS-TE MIB had grown significantly in both size and scope. The primary reason for this was due to feedback and requirements from those deploying the MIB. In essence, the MIB has grown to encompass the functionality of both general LSR functions as well as traffic engineering functions. It was at this point that the MPLS Working Group decided that the MPLS-TE-MIB needed to be split into two MIBs: one to encompass general LSR switching functions, and one to encompass the general MPLS traffic engineering capabilities. It was at this point that the MPLS Label Switching Router MIB (MPLS-LSR MIB) was split from the MPLS-TE MIB and chartered as a separate Working Group item.

As time went on, the feedback process from service providers continued. It was at this time that the MPLS FEC-to-Next-Hop Label Forwarding Entry MIB (MPLS-FTN MIB) was proposed to expose the FEC-to-NHLFE mapping within LERs. In addition, MPLS BGP/VPNs were proposed, and implementation of this new application of MPLS had begun as well. Not long after this, the Provider-Provisioned MPLS Virtual Private Network (PPVPN-MPLS-VPN MIB) was proposed to the IETF and was adopted. It is likely that there will be many other standard MIBs provided by the IETF that cover all of the essential and common MPLS functionality, thereby making the manageability of MPLS networks far easier and straightforward for those who choose to utilize this technology.

Table 1.2 MPLS MIB module drafts.

MIB title	Date started	Current status	Description
draft-ietf-mpls-ldp-mib-*.txt	August 1998	Currently under IESG review	LDP protocol
draft-ietf-mpls-te-mib-*.txt	November 1998	Currently under IESG review	Traffic engineering
draft-ietf-mpls-lsr-mib-*.txt	June 1999	Currently under IESG review	Active TFIB of an LSR
draft-ietf-mpls-ftp-mib-*.txt	November 2000	MPLS WG last call.	FEC-to-NHLFE mapping
draft-ietf-ppvpn-mpls-vpn-mib-*.txt	August 2001	Adopted as Working Group draft by PPVPN WG	MPLS VPN
draft-nadeau-mpls-ds-te-mib-00.txt	February 2001	Awaiting adoption by TE WG	Differentiated Services-aware traffic engineering
draft-ietf-mpls-tc-mib-*.txt	June 2001	Under IESG review	Common textual conventions for MPLS MIBs
draft-nadeau-gmpls-te-mib-*.txt	March 2002	Adopted by CCAMP WG	GMPLS TE
draft-nadeau-gmpls-tc-mib-*.txt	March 2002	Adopted by CCAMP WG	Common textual conventions for GMPLS MIBs

Table 1.2 enumerates many of the MIB modules that are available at the time this text was written. More will surely become available as time goes on. However, we provide this table to illustrate those that are currently available so that you might see the progression from essentially no standard management interfaces for MPLS, to the near dozen available today.

The remainder of this text will focus on SNMP MIB-based solutions for managing MPLS networks. To that end, Figure 1.10 illustrates how each MIB fits in with the others as well as how each one depends on the others. This illustration should be a guide for you on how you wish to pursue each chapter. The book is presented in an order such that the later MIBs depend on the earlier ones. The MIBs presented earlier generally do not have dependencies on the ones that come after them.

This order presents a road map of sorts regardless of whether you are reading the text from an operational or a developmental viewpoint. That is, we recommend that you examine the MIBs presented first before tackling those presented later in the text, as they form a foundation not only for inherited or reused MIB objects, but also for concepts that are used throughout the MIBs, but may be introduced in one of the earlier MIBs.

The MIBs are organized as follows. The MPLS-TC MIB describes textual conventions that are used by all MPLS-related MIBs (even ones too new to be covered at this time). A version of the TC MIB that existed at the time of publication of this book is included in Appendix B. Note that it may have changed to an RFC version, as it was being reviewed at that time. The remaining four MIBs are shown as having dependencies on the MPLS-TC MIB, as well as the IF-MIB (RFC 2863). Since nearly all of the MIBs are related to the IF-MIB, a specific icon has not been included for it in the figure; instead, a gray triangle in the corner of each MIB indicates this dependency.

The MPLS-LSR MIB (see Chapter 3) describes the basic label forwarding operations of an LSR. The MPLS-LSR MIB also exposes which interfaces the LSR has MPLS enabled on by cross-referencing each MPLS-enabled interface that appears in the IF-MIB. This MIB presents a foundation of actual objects (as opposed to TCs in the MPLS-TC MIB) that are used in many other MIBs; thus it is viewed as the base MPLS MIB by many.

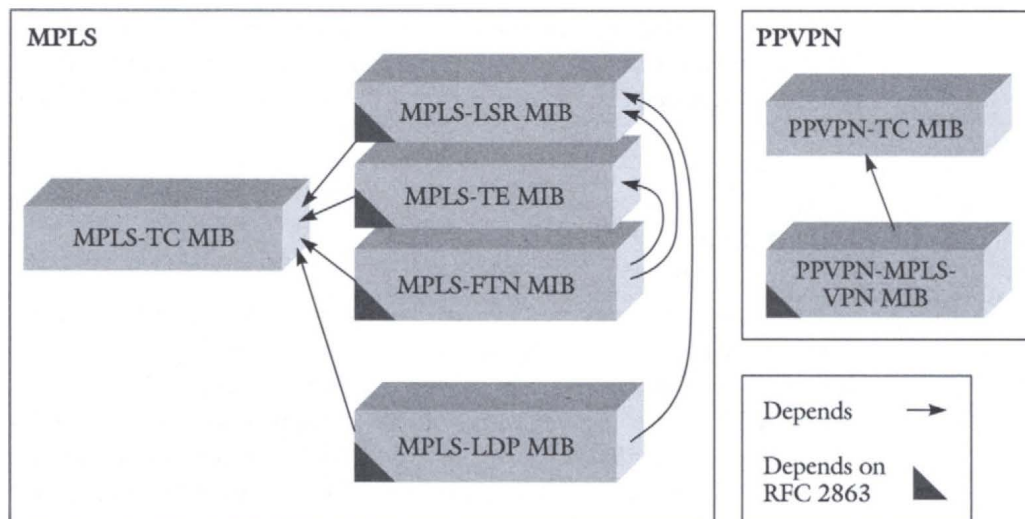


Figure 1.10 MIBs for MPLS network management discussed in this text.

The MPLS-TE MIB provides the operator with a view of which traffic engineering tunnels are configured, signaled, or presignaled (for backup). If a tunnel is also represented as an interface in the IF-MIB, an entry will exist there as well. The MPLS-TE MIB (see Chapter 8) depends on the MPLS-LSR MIB in that the system software in a device can be programmed to associate the active LSP with a tunnel when such a relationship exists.

Next, the MPLS-LDP MIB (see Chapter 4) provides insight into what the LDP protocol is doing on an LSR, assuming that LDP is enabled and in use. The MPLS-LDP MIB depends on the MPLS-LSR MIB for its mapping tables that are used to associate LDP sessions with active LSPs. The MPLS-LDP MIB also depends on the IF-MIB in that it exposes which label ranges are configured on an MPLS-enabled interface. Finally, the MPLS-FTN MIB (see Chapter 5) presents the operator with a view of how IP traffic is entering the MPLS network and how that IP traffic is being mapped onto MPLS LSPs or traffic-engineered tunnel interfaces.

The MPLS-FTN MIB depends on the MPLS-LSR and MPLS-TE MIBs because the way that it associates incoming IP traffic is to point at the associated LSP or traffic engineering tunnel head as represented in the MPLS-LSR and MPLS-TE MIBs, respectively. The MPLS-FTN MIB depends on the IF-MIB because it allows an operator to configure FEC-to-NHLFE mapping rules on a per-interface basis.

The PPVPN-MPLS-VPN MIB is shown to possess only dependencies on the PPVPN-TC MIB. This MIB contains common textual conventions used by the PPVPN-MPLS-VPN MIB as well as other MIBs defined by the IETF PPVPN Working Group. The PPVPN-MPLS-VPN MIB provides an operator with a view of which VPN instances are configured on a specific PE, as well as related statistics, BGP, and interface information. The interface information extends those interfaces that are already represented in the IF-MIB; thus yet another dependency on the IF-MIB exists. A detailed discussion of the PPVPN-MPLS-VPN MIB can be found in Chapter 11.

1.5 Summary

This chapter covered the basic components of Multi-Protocol Label Switching (MPLS). At the heart of MPLS is the separation of the control and forwarding planes. There are distinct advantages to this approach, as we have seen. The forwarding plane is composed of various IP-based protocols such as BGP and RSVP. The forwarding plane is based on switching a short, fixed-length label. This method is based on the forwarding mechanisms of several layer-2 forwarding technologies such as ATM and Frame Relay.

The remainder of the chapter introduced some of the reasons why it is crucial for world-class MPLS deployments to provide robust and comprehensive network management capabilities. We discussed fault and configuration management and how these two components of a management solution alone were critical if the network was to be deployed effectively, especially on a large scale. We then discussed performance measurement within the context of service level agreements. Monitoring and verifying the quality of a network connection is especially important when required by a service level agreement.

Finally, we presented an overview of how the MPLS MIBs fit together to provide the reader with a pictorial “30,000 view” of the MIBs that will be discussed later in the text. A current overview of how the MPLS MIBs interrelate can be found in Nadeau et al. (2001). We will refer to this figure throughout the book. Due to the large number of MIBs covered, we will repeat Figure 1–10 at the beginning of each chapter covering a particular MIB. In the opening of each chapter we will highlight the MIB that will be discussed to refresh your memory of where it fits into the larger picture of MPLS management.

It was not our goal for this chapter to be an exhaustive introduction to MPLS, since that is not the focus of this text. A comprehensive introduction is not in the scope of this book. For a more all-inclusive understanding of the basics of MPLS, please refer to the books listed in the Further Reading section. You are also invited to follow the ongoing work in the IETF via their Web site listed in the Further Reading section. Another way to follow the IETF work closely is by attending its meetings or by following the relevant discussions on its various mailing lists.

Further Reading

Davie, B. S., and Y. Rekhter. *MPLS: Technology and Applications*. First edition. San Francisco: Morgan Kaufmann Publishers. 2000.

Gray, E. W. *MPLS: Implementing the Technology*. Reading, Mass.: Addison-Wesley Professional. 2001.

To find out more about the IETF, visit their Web page at www.ietf.org/.

For more information about the Internet Assigned Numbers Authority (IANA), check out their Web site at www.iana.org/.

To locate information about the ITU: www.itu.int/.

To locate information about the MPLS Forum: www.mplsforum.org/.

To locate additional information about the OIF: www.oiforum.com/.

Cisco’s Web site also provides a great deal of information regarding MPLS: www.cisco.com.

George Swallow

is the co-chair of the IETF's Multi-Protocol Label Switching Working Group. He is also a technical leader at Cisco Systems, where he is a member of the architecture team for label switching. He is the chief architect for MPLS traffic engineering. He is now involved in extending the MPLS control plane into the optical space. He is also Cisco's technical voting representative to the Optical Internetworking Forum.

George, you have been an active member of the IETF community since the early days of MPLS. At that time, people were concerned with early deployment issues such as just getting the technology working. However, today MPLS is a mature technology and has been deployed in numerous production networks. What do you see as the issues and hurdles (and possible solutions) for current and future deployments of MPLS?

As you point out, MPLS is a mature technology. Like any mature technology, its future deployment is driven largely by economics. As a technology, MPLS is not an end unto itself; the economics of MPLS are driven by the applications MPLS enables.

Let's speak of four kinds of applications:

1. Virtual private networks
2. Transport of legacy services
3. Traffic engineering
4. Quality of service

Note that these are not mutually exclusive and are often used in combination.

Virtual private networks: As IP comes to dominate data networks, it becomes economical for service providers and large corporate networks to combine their offering on a single backbone. VPNs offer a means of not only moving the customer base from a Frame Relay or ATM network, but also enable the service provider to offer directly what most customers desire, namely, secure outsourced IP connectivity.

Transport of legacy services: The economics here are similar to those for VPNs, except that one is offering only legacy service, Frame Relay, ATM, and in some cases SONET/SDH. Again, as IP dominates, it will at some point cease to be economically viable to maintain separate network(s) to offer the above service. MPLS allows these services to be offered over the IP backbone.

Traffic engineering: On a strategic basis, traffic engineering is an adjunct to network design. You design a network in anticipation of future traffic loads. In IP networks, you only have so much control over which path through a network a particular flow will take. Traffic engineering allows much tighter control over the mapping for flows to paths. Like any optimization problem, such as packing boxes, you can reach more optimal solutions if you have smaller items to put in the boxes. Serious bandwidth savings can be achieved. But in a time where there is a surfeit of bandwidth, the operational cost of engineering traffic cannot be justified in many cases. On a tactical basis, traffic engineering can be used to engineer traffic around network "hot spots" and to optimize the use of expensive transmission facilities such as transoceanic links. This allows a provider to allow a better level of service with the current network design. Tactical traffic engineering is particularly useful in dealing with unanticipated traffic growth or changes in usage patterns and with unplanned outages of network facilities. It seems to me that, even in the current economic climate, tactical use of traffic engineering makes sense.

Quality of service: I hesitate to guess here. QoS has been talked about for a long time. What I will say is that I believe it will be application-driven. At some point, it will make sense to push a lot of voice onto campus and wide-area backbone networks. I think that this will be the time when we really see QoS come to the fore. Note that MPLS can provide not only the QoS needed by voice (and other real-time applications such as video), but also fast restoration. This is a feature closely related to traffic engineering that we haven't spoken about. Essentially MPLS fast restoration offers IP the kind of restoration times found in SONET, allowing critical applications to suffer no more than a blip in the face of a link or node failure.

Some people feel that offline traffic engineering is the only way to correctly engineer and maintain an MPLS network. What is your take and advice in this area? Where do you see offline TE as fitting into a complete strategy for managing an MPLS network today, and in the future for a GMPLS network?

First, traffic engineering is not necessary for an MPLS or any other network. Congestion problems can always be solved by throwing more bandwidth at the problem. As I said above, you need to differentiate between strategic and tactical traffic engineering. In strategic traffic engineering you seek to essentially engineer all of the traffic in the network. The process is very tightly coupled to the process of network design. Like network design, good offline tools are needed in order to tackle a network of considerable size. In the tactical case, you try to engineer as little traffic as is necessary to adequately relieve some network hot spot. Here offline tools, though still useful, are not necessary.