MPLS Virtual Private Networks

Luca Cittadini Giuseppe Di Battista Maurizio Patrignani

Summary

This chapter is devoted to Virtual Private Networks (VPNs) designed with Multi Protocol Label Switching (MPLS) [14, 15, 1], one of the most elusive protocols of the network stack. Saying that MPLS is "elusive" is not overemphasizing: starting from its arduous fitting within the ISO/OSI protocol stack, continuing with its entangled relationships with several other routing and forwarding protocols (IP, OSPF, MP-BGP, just to name a few), and ending with the complex technicalities involved in its configuration, MPLS defies classifications and challenges easy descriptions.

On the other hand, and in a seemingly contradictory way, the configuration of VPNs with MPLS is rather simple and elegant, despite the complexity of the underlying architecture. Also, MPLS flexibility and maintenance ease make it a powerful tool, and account for its ubiquity in Internet Service Providers' networks.

The chapter is organized as follows. Section 1 gives a brief introduction and motivation behind the concept of Virtual Private Network and explains why Layer 3 MPLS VPNs are by far the most popular widespread kind of VPNs deployed today.

In Section 2 we introduce the reader to basic concept and terminology about Label Switching (also known as Label Swapping) and Virtual Private Networks.

Section 3 gives a high-level step-by-step description of an MPLS VPN. This is based on three main ingredients: an any-to-any IP connectivity inside the network, a signalling mechanism to announce customer IP prefixes, and an encapsulation mechanism, based on MPLS, to transport packets across the network.

Section 4 explores in detail the complex interplay between IP and MPLS that is at the basis of MPLS VPNs.

More technical details about dynamic routing and connecting to the Internet, advanced usage of routing, and preserving IP-specific per-hop behavior are provided in Section 5.

Strengths and limitations of MPLS VPNs are discussed in Section 6. The same section proposes further readings on the subject.

The reader who is interested in getting only a high-level understanding on how MPLS VPNs work can read Sections 1, 2, and 3. An indepth view of MPLS VPNs can be gained by reading Sections 4 and 5.

1 Virtual Private Networks

After giving a brief introduction and motivation behind the concept of Virtual Private Network, this section explains why Layer 3 MPLS VPNs are by far the most popular widespread kind of VPNs deployed today.

L. Cittadini, G. Di Battista, M. Patrignani, "MPLS Virtual Private Networks", in H. Haddadi, O. Bonaventure (Eds.), *Recent Advances in Networking*, (2013), pp. 275-304.

1.1 The Need for Virtual Private Networks

The concept of Virtual Private Networks (VPNs) is essential in today's networks and will probably become paramount in tomorrow's networks, yet it is sometimes considered too advanced to be covered in a networking course. This apparently contrasts with the simplicity of the concept of a VPN: in its most generic form, a VPN is a closed ("Private") group of nodes that want to be connected in a network ("Network") and are willing to use virtual connections, or *pseudowires* ("Virtual") instead of physical connections.

Such a definition captures the essence of a VPN from the perspective of the customer. A network provider has a slightly different abstraction about a VPN, mostly because she has a different interpretation of the keyword "Network": within the graph that represents her own network, she needs to provide connectivity to a subset of the nodes. Despite being seemingly very easy, each of the other two keywords that appear in the definition hides a fair amount of complexity that is not obvious at first glance.

Virtual Where in the ISO/OSI stack does virtualisation happen?

Private Is there any authentication mechanism? Does the VPN need to preserve confidentiality of the messages?

Each of these questions has many possible answers, which is the reason why there are so many different types of VPNs in today's networks. For example, a peer-to-peer network can be seen as a VPN where pseudowires are transport sessions, there is no authentication amongst nodes and no traffic encryption, and the topology of the network is defined by a dynamic algorithm. At the opposite side of the spectrum we have optical networks, which can be seen as VPNs where pseudowires are light paths through optical switching devices, there is no authentication and no encryption, and the network topology is defined by simply configuring arbitrary pseudowires among the nodes.

In the context of VPNs, the term "virtualisation" indicates the technology that is used to multiplex traffic from multiple VPNs on the same underlying network topology. The most important feature of a VPN technology is what multiplexing technique is used and at which layer of the protocol stack. In general, pushing the multiplexing component down to the lower layers of the protocol stack (e.g., the physical or data-link layer) implies a higher implementation cost compared to the higher layers (e.g., the transport or application layer). For example, deploying an optical network to be able to run arbitrary pseudowires between two computers is several orders of magnitude more expensive than connecting those two computers to the Internet and writing a software that establishes a tunnel between them. On the other hand, multiplexing is transparent to upper layer protocols: for this reason, multiplexing at lower layers in the stack allows us to support a wider fraction of the protocol stack.

The most common layers where multiplexing happens are layer 2 and layer 3. A layer 2 VPN (L2VPN) transports packets of a specific layer 2 protocol and hence, thanks to the layered architecture of the protocol stack, is capable of supporting any kind of layer 3 protocol. L2VPN technologies join the nodes belonging to the same VPN within the same broadcast domain. For example, with a L2VPN, all nodes in the VPN could participate in the same VLAN and exchange Ethernet packets. We refer the reader to [3] for a detailed discussion of requirements for L2VPNs, and to [2] for a reference model of L2VPNs and a discussion of the main functional components. Analogously, a layer 3 VPN (L3VPN) transports packets of a specific layer 3 protocol and hence is capable of supporting any kind of layer 4 protocol. Nodes belonging to the same L3VPN can exchange IP packets that are routed through a provider network. We refer the reader to [8] for a detailed discussion of requirements for L3VPNs, and to [7] for a reference model of L3VPNs and a discussion of the main functional components.

1.2 Layer 3 VPNs and MPLS

Layer 3 VPNs are by far the most popular of VPNs deployed today. One reason is that layer 3 offers a good trade-off between deployment cost and transparency to end hosts. Another, perhaps stronger reason is that, as the Internet converged towards today's everything-over-IP scheme, it seemed natural to place the multiplexing component at the highest layer that supports transporting IP packets[1].

Despite a variety of technologies to realize virtual layer 3 services, most L3VPNs are based on the Multi Protocol Label Switching protocol (MPLS). The popularity of an L3VPN technology strongly depends on its ability to meet the demands of customers, providers, and vendors:

Customers' needs: Typical VPN customers (e.g., private companies, public administrations, etc.) have several geographically distributed sites and would like to have a unique IP network connecting all of them. Besides mere connectivity, they have other requirements: (i) they want to keep their own IP addressing plan for all the sites; (ii) they want their traffic to be logically separated from the traffic of other customers that happen to use the same shared infrastructure; and (iii) they want guaranteed quality of service.

Providers' targets: Providers have invested lots of resources in building their own network backbone. Since they have an existing infrastructure with many distributed PoPs (Points of Presence) connected to the backbone, they would prefer to sell pseudowires rather than physical connections to their customers. Among multiple techniques to implement pseudowires, providers prefer those that involve lower configuration efforts, which usually implies lower maintenance costs. Moreover, they want the implementation to be scalable with respect to the number of customers: the amount of state to keep in the core of the network should only depend on the network topology, not on the number of customer VPNs.

Vendors' strategies: No network technology can be easily deployed without meeting the strategies of network device producers and vendors, whose immediate aim is to sell many machines (possibly expensive carrier-grade routers) and, in the long run, to drive the shift from a variety of old technologies for VPNs (e.g., ATM or Frame Relay) to new technologies that are simpler to manage and hence have the potential to grow the vendor's market share.

After having introduced MPLS terminology and having given an overview of its main building blocks, in Section 6 we discuss the extent to which MPLS is able to meet the above requirements.

Throughout this chapter we will refer to a very simple scenario (see Fig. 1) where a provider has a network infrastructure with three PoPs (in Turin, Milan, and Rome) and offers connectivity to two customers. Customer 1 has two sites and has an IP addressing plan that allocates the $200.1.6.0/24$ to its site in Milan and the $10.0.0.0/24$ to its site in Rome. Customer 2 has two sites too and has an IP addressing plan that allocates the $10.0.0.0/24$ to its site in Turin and the $193.1.1.0/24$ to its site in Rome. Observe that the two customers have overlapping IP address space.

We will use the sample scenario to illustrate most of the concepts introduced in this chapter. The text that describes and refers to the sample scenario will be framed into shaded boxes like the one that encloses this paragraph. The configuration language is that of a leading router vendor and references can be found in [9].

[1] We are recently observing a similar convergence trend at layer 2 with Ethernet: consequently, in the last few years there has been a significant increase in the demand of virtual layer 2 services.

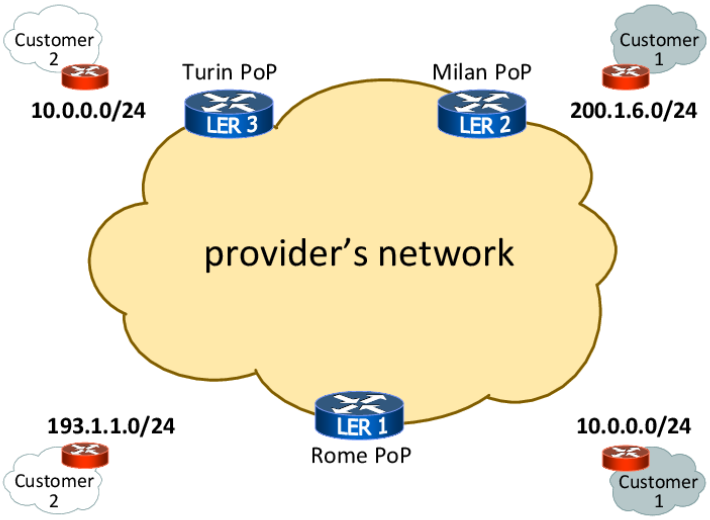

Figure 1: The sample network used throughout this chapter.

2 Background and Terminology

In this section we introduce the reader to basic concept and terminology about Label Switching (also known as Label Swapping) and Virtual Private Networks.

Throughout the chapter, we extensively refer to two tightly related yet distinct concepts: forwarding and routing. *Forwarding* is the process of receiving a packet from a network interface and deciding on which interface that packet should be sent. Usually, in order to minimize the latency of traversing a router, the decision about where to forward a packet is taken based on some pre-computed data structure. This is usually referred to as the *forwarding table* because a table is the simplest logical structure to accommodate forwarding information. A table can be used as a physical data structure if addresses can be matched exactly. However, IP addresses must be forwarded based on the longest matching prefix [10]. This implies that efficient IP lookups need a more sophisticated data structure than a table. We refer the reader to [18] for a discussion on various data structures and algorithms to speed up prefix-match lookups, which exceeds the scope of this chapter. In the following, we simply refer to the logical forwarding table, irrespective of the actual physical data structure.

Routing is the process by which each router builds its forwarding table and adapts it as the network topology changes over time.

Correspondingly, we have *forwarding and routing protocols*, where the formers describe the formatting rules for network packets and the conventions that routers and hosts have to follow in order to exchange them, while the latter describe packet formats and conventions used to exchange routing information among routers. The information that routing protocols provide is used by each router to populate its forwarding table.

Finally, standard network terminology distinguishes between the corresponding router's software layers. Namely, the layer where the forwarding process takes place is called *data plane* or *forwarding plane*, while the layer where the routing process is managed is called *control plane*.

Destination address	Egress interface
10.100.100.0/24	en2
10.100.200.128/25	en1
22.30.100.0/24	en2

Table 1: Structure of the forwarding table in the "forwarding by network address" approach.

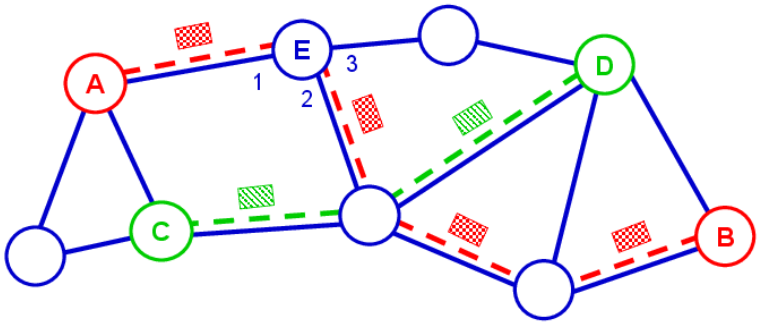

Figure 2: A label switching network (where labels are not swapped at each hop).

2.1 Label Switching

In this section we introduce the concept of label switching as a forwarding paradigm. After having described the fundamentals characteristics of label switching in general, we move to MPLS-specific details in the following sections. Traditionally, there are two different approaches to packet forwarding, each mapping to a specific structure of the forwarding table. They are called *forwarding by network address* and *label switching*.

The most intuitive approach is *forwarding by network address*, that is the approach of IP. When a packet arrives at a router, the router parses the destination address from the packet header and looks it up in its forwarding table. The forwarding table has a simple 2-column structure where each row maps a destination address to the egress interface that the packet should be forwarded to (see Table 1). For scalability and efficiency reasons, it is possible to aggregate several destination prefixes into a single row, provided that they can be numerically aggregated and that they share the same egress interface.

An alternative approach is known as *label switching*. Essentially, while forwarding by network address requires that the egress interface be chosen based on the *destination* of the packet, label switching requires that such an interface be chosen based on the *flow* the packet belongs to, where a flow corresponds to an instance of transmission, i.e., a set of packets, from a source to a destination and is identified by a tag (called *label*) attached to each packet of the flow.

As an example, Fig. 2 shows a label switching network where each flow has an associated label (labels

Incoming interface	Incoming label	Egress interface	Egress label
en2	101	en5	218

Table 2: Structure of the forwarding table in the "forwarding by label swapping" approach with per-interface label scope.

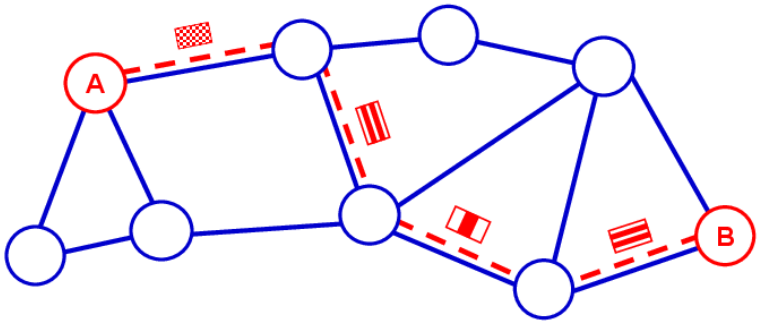

Figure 3: A label switching network (where labels are swapped at each hop).

Incoming label	Egress interface	Egress label
101	en5	218

Table 3: Structure of the forwarding table in the "forwarding by label swapping" approach with per-router label scope.

are represented with colors). Packets with a red label belong to the flow from router A to router B of Fig. 2. If a packet with a red label enters interface 1 of router E, it will exit from interface 2 with the same label.

If the label switching technologies followed the approach of Fig. 2 they would have the advantage that labels do not have to be changed at each hop. On the other hand, if they did they would have the big drawback of requiring a centralized control of the assigned labels, as labels should be unique for the entire network. Fig. 3 shows what actually happens in label switching networks, where labels are swapped at each hop. This choice requires that labels are unique for each router or for each interface only and does not need a centralized control. Fig. 4 illustrates the forwarding table of a router. Independent of whether labels are swapped at each hop or not, the forwarding paths towards the same destination node typically form a tree rooted at the destination node itself, even though this is not mandatory.

More formally, the operations performed by a label switching router can be summarized as follows. When the packet arrives at the router, the router extracts (*pops*) the label from the header, looks the label value up in its forwarding table, and finds (i) the egress interface the packet should be forwarded to, and (ii) a new label to apply (*push*) to the packet.

A forwarding process based on labels rather than destination addresses poses challenges to the corresponding routing protocols. In fact, the instances of flow traversing the network might be much more volatile than the addressing scheme used to identify their destinations. Before transmitting a new flow, a route from its source to its destination has to be computed and a new label has to be assigned to each leg of the route. As observed before, in order to facilitate the task of picking a new, unused, label, labels are not required to be unique for the entire network but are required to be unique for each router or for each interface only. This is why they have to be changed at each hop. Depending on whether labels have a per-interface or per-router scope, the forwarding table is structured as in Table 2 or Table 3, respectively. Observe that such a simple structure for forwarding tables allows efficient lookups (e.g., by using a hash table or a direct access table).

Label switching is not a unique feature of MPLS and it is not necessarily implemented at the network level of the protocol stack: other protocols, notably ATM and Frame Relay, traditionally adopt the same

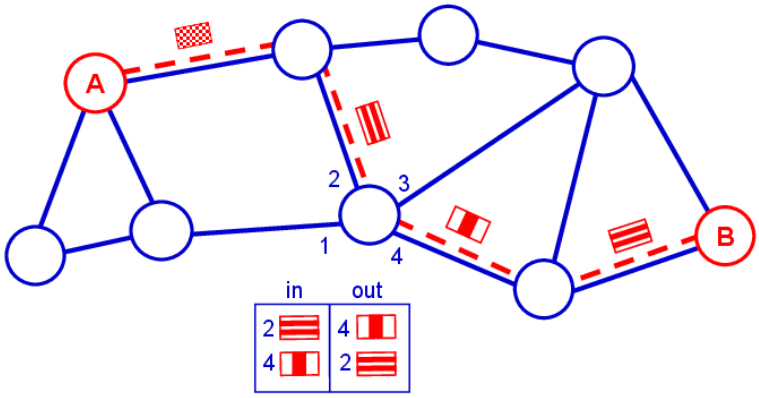

Figure 4: A label switching network where the forwarding table of a router is shown.

Layer 3	IP
Layer 2.5	MPLS
Layer 2	Ethernet, Frame relay, ATM, PPP, etc
Layer 1	Physical layer

Figure 5: MPLS and ISO/OSI network layers.

forwarding mechanism. Initially, the reason to prefer label switching was performance: looking up a label value in the forwarding table was much faster than looking up an IP address. Besides the fact that labels can take values in a much smaller range than IP addresses, label values can be looked up exactly, while IP addresses need to be looked up by the longest matching prefix. However, modern routers use extremely specialized hardware (e.g., content-addressable memories) and very efficient data structures (e.g., tries) to implement their forwarding tables, in such a way that the performance gain of label switching over forwarding by destination address is now believed to be no longer an argument.

2.2 MPLS header and terminology

The MPLS protocol brings the label switching forwarding paradigm to the extreme, managing, instead of a single label, a whole stack of labels, where the external one determines the egress interface. It does not fit the ISO/OSI model very well. The MPLS header is transported over L2 packets and can encapsulate L3 packets as well as L2 packets. Since MPLS does not fit the definition of either L2 protocols nor L3 protocols, it is frequently referred to as a "layer 2.5" protocol, emphasizing the fact that it requires L2 connectivity and can encapsulate IP packets.

When an IP packet from a router needs to be transported over an MPLS backbone, the first MPLS-enabled router in the network pushes an MPLS header in between the Ethernet header and the IP header. The resulting packet layout is depicted in Fig. 5.

The MPLS header consists of a *stack* of 4-byte records where each record has the following structure (depicted in Fig. 6):

- a **label** field (20 bits), which carries the label value;

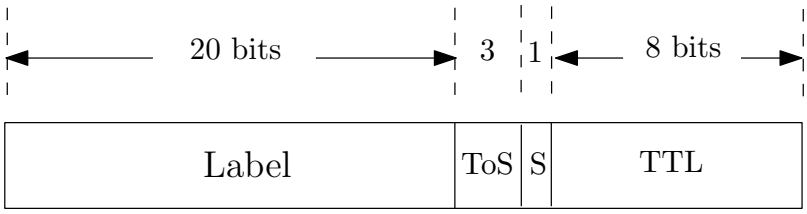

Figure 6: Structure of a record in an MPLS header.

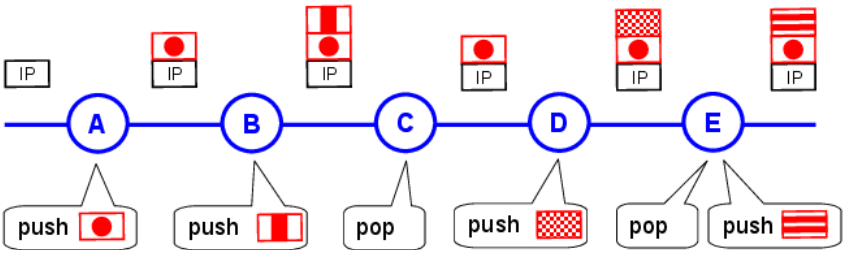

Figure 7: The evolution of the MPLS label stack as a packet traverses several routers that perfom random push and pop operations.

- a **ToS** field (3 bits) which is used to discriminate different levels of quality of service (QoS) and to carry explicit congestion notifications (ECN);
- a **bottom-of-stack** field (1 bit) which is set to 1 when the record is the last record in the stack; and
- a **TTL** field (8 bits) which is decremented at each hop, similarly to the TTL field in the IP header.

When an MPLS-enabled router receives a packet, it can perform three different operations: (i) *push* a label onto a (possibly empty) stack, (ii) *pop* a label from the stack (possibly resulting in an empty stack), or (iii) *swap* the top label of the stack, which can be seen as a pop operation followed by a push operation. Figure 7 shows the evolution of the MPLS label stack as a packet traverses several routers that perform random push and pop operations.

MPLS-VPN terminology uses specific names to distinguish routers that do not understand labels at all, routers that push (or pop) labels, and routers that simply swap labels. Routers belonging to the first group are called *customer edge* (CE) routers because they are not MPLS-enabled. Typically those are the customer's routers that need to be interconnected via an L3VPN. CE routers can only handle IP packets and are not aware of the MPLS layer which is used to implement the VPN.

Routers belonging to the second group are called *provider edge* (PE) routers, or *label edge routers* (LERs). They are placed at the edge of the MPLS backbone of the provider, have direct connectivity to the CE routers, and act as the access point for the customer to the VPN. While they need to perform label swap operations because they are part of the backbone, they spend most of their time pushing labels (when an IP packet comes from a CE router) and popping labels (when an MPLS packet needs to be forwarded to a CE router).

Routers belonging to the third group are called *provider* (P) routers, or *label switching routers* (LSRs). They are in the core of the MPLS network. Since they do not interact directly with non-MPLS routers, they

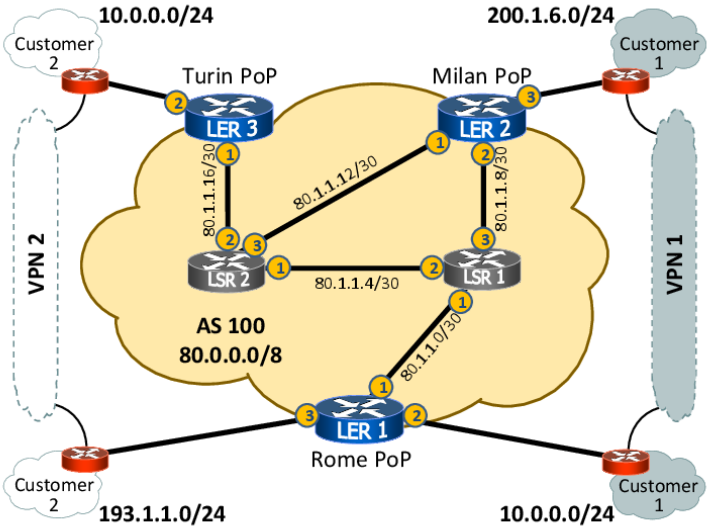

Figure 8: Inside the provider's infrastructure.

mainly perform label swapping operations in order to forward packets to other P or PE routers.

MPLS groups destinations into Forwarding Equivalence Classes (FEC). Packets that need to be forwarded to the same CE using the same path and with the same quality of service belong to the same FEC.

Fig. 8 shows some details of the provider's infrastructure of our scenario. It is both an MPLS network and an IP network (it has an MPLS data plane and an IP data plane).

If we look at it from the MPLS point of view, we can distinguish CE, PE, and P routers. The small, red routers placed at the customer premise in the corners of Fig. 8 are CE routers. CE routers are directly attached to the blue routers at the edge of the provider premise, which are the PE routers (or LERs). Finally, the grey routers in the core of the provider network are the P routers (or LSRs).

Since the provider network is also an IP network an IP address is given to the interfaces. To do this, our provider exploits prefix $80.0.0.0/8$. This prefix will not be announced outside the provider's network. The reason for the presence of label AS100 in the provider network will be explained soon.

The two CE routers serving Customer 1 are connected through a VPN called VPN1 (on the right side of Fig. 8). The two CE routers serving Customer 2 are connected through a VPN called VPN2 (on the left side).

3 Checkmate VPNs in Three Moves

In this section we give a high-level description of an MPLS VPN. Such a description is based on three main ingredients that we call "moves". We claim that a reader that understands these three moves will be able to checkmate this complex matter.

From the perspective of the customer, an MPLS VPN simply routes IP packets among customers CE routers, as if they were connected by a *pseudowire*. Observe that, as customers may have overlapping

address spaces, their packets cannot be simply routed through the provider network. Instead, they need to be encapsulated. It is tempting to implement such a pseudowire using a tunnel (e.g., GRE, IP-in-IP or IPSec) between PE routers where the customer packets travel across the provider network *encapsulated* into IP or IPSec packets. However, as the number of interconnected sites grows, manually managing configured tunnels and maintaining forwarding tables becomes excessively complex. For example, if we were to use tunnels to implement an L3VPN over 5 customer sites, a full-mesh topology would translate to 20 manually configured tunnels. Moreover, if the customer adds a new subnet to one of its sites, we need to update the forwarding tables of all our 5 PE routers. Observe that the complexity of managing tunnels can be eased by automatic setup mechanisms. However, such mechanisms are out of the scope of this chapter, therefore we refer the interested reader to [16, 13, 17].

The intrinsic problem with tunnels is that they rely on a pre-determined endpoint which is configured at tunnel setup time. Ideally, we would like to take advantage of the benefits of encapsulation without dealing with the issue of knowing the tunnel endpoint in advance. Namely, we would like packets to be encapsulated at the ingress PE and decapsulated at egress PE. We can split this goal into three high-level steps that we call moves:

Move 1: Achieve any-to-any IP connectivity among PEs,

Move 2: Define a signalling mechanism to distribute customer prefixes among PEs, and

Move 3: Define an encapsulation mechanism to transport packets from one PE to another across the network.

One of the key benefits of using encapsulation (Move 3) is that the complexity of configuring L3VPNs for customers is confined to PEs. The core of the network (i.e., P routers) does not need to know anything about customer prefixes: it simply needs to know how to transport packets from one PE to another (Move 1). This means that the size of the forwarding table of P routers depends on the number of PE routers rather than on the number of customer prefixes. Finally, if PE routers use a signalling mechanism to dynamically synchronize the list of customer prefixes, the only pieces of information that need to be manually configured at each PE are the L3VPN identifier and the IP address of the CE router.

In the following we elaborate each move in more detail.

3.1 Move 1 – Any-to-any IP connectivity among PEs

The first move is actually quite simple. It is nothing more than what any Internal Gateway Protocol (IGP) is designed to achieve: seamless, redundant and dynamic IP-level any-to-any connectivity. Since PEs are our encapsulation endpoints, we want them to be reachable independent of the availability of specific network interfaces. In other words, we do not want to use the IP address of physical interfaces for PEs, but loopback addresses. A loopback address is an address associated with a virtual interface of the router. Since it is virtual, a loopback interface it is active independent of the status of physical network interfaces. To fulfill Move 1, we simply assign a loopback address to each PE router and use an IGP (e.g. OSPF or IS-IS) to announce these addresses as /32 prefixes in order to ensure any-to-any connectivity among them.

Fig. 9 shows the loopback addresses assigned to the PEs of our example network. Also, we assume that routers use OSPF to propagate reachability information of loopbacks of routers.

Configuring routers to fulfill Move 1 is straightforward. In our sample network, the configuration of LER1 for Move 1 is as follows.

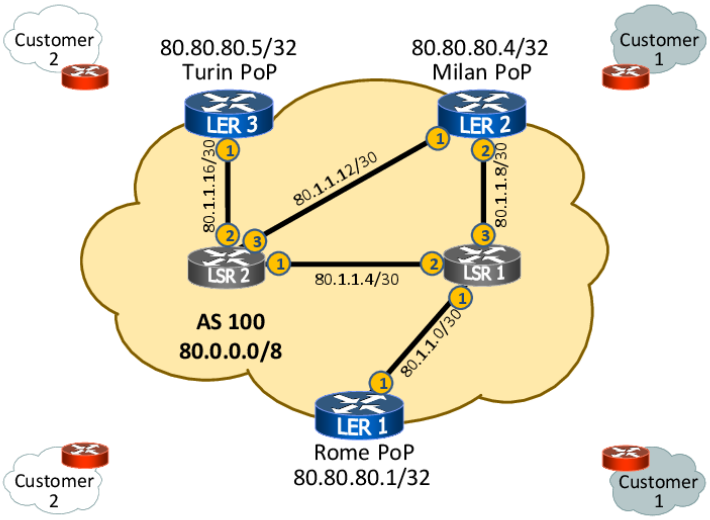

Figure 9: Loopbacks of PEs.

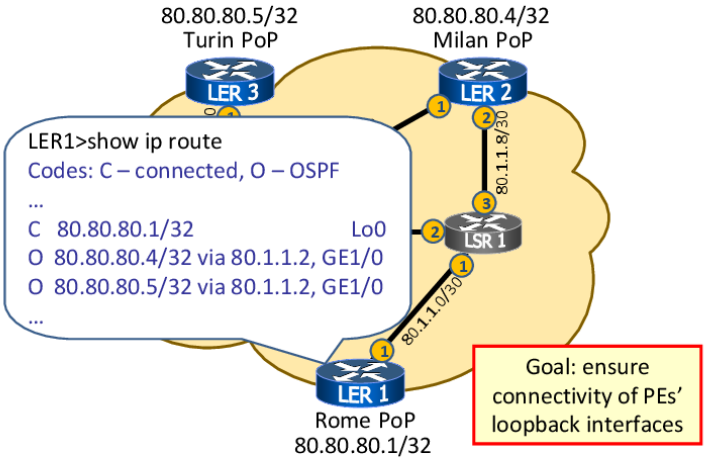

Figure 10: IP connectivity for LER1.

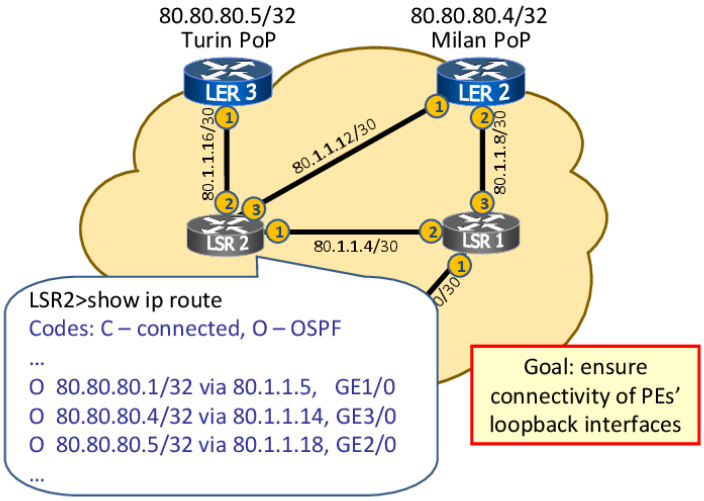

Figure 11: IP connectivity for LSR2.

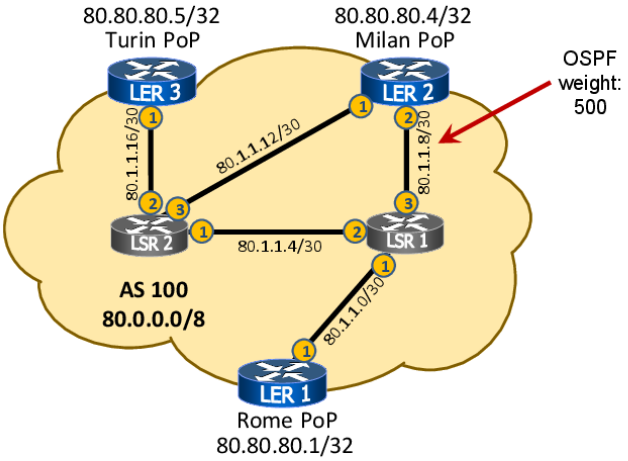

Figure 12: The OSPF weight of a link.

```
interface Loopback0
  ip address 80.80.80.1 255.255.255.255
interface GigabitEthernet1/0
  ip address 80.1.1.1 255.255.255.252
router ospf 10
  network 80.0.0.0 0.255.255.255 area 0
```

The first two lines assign an IP address to interface `loopback0`. The second pair of lines assign an IP address to interface `GigabitEthernet1/0` that connects LER1 with LSR1. The last two lines activate OSPF protocol.

Fig. 10 shows the result of command `show ip route` performed on router LER1. Fig. 11 shows the result of command `show ip route` performed on router LSR2. Command `show ip route` has the effect of showing the control plane routing table of routers.

In order to force a more interesting routing in the following part of the example, we set OSPF weight 500 for a specific link, discouraging the use of that link by the IGP routing protocol, as shown in Fig. 12.

3.2 Move 2 – Use BGP to distribute customer prefixes

In order to distribute reachability information about customer prefixes, MPLS relies on a variant of BGP called Multi-Protocol BGP (MP-BGP)[5]. Whereas BGP advertises reachability information for IPv4 addresses only, MP-BGP supports multiple *address families* (e.g., IPv4 and IPv6). Since advertising VPN addresses implies exchanging not only IPv4 prefixes, but also additional information to identify the VPN, MP-BGP treats VPNs as a separate address family. PE routers establish a full-mesh of iBGP peerings and each PE announces to all the other PEs the customer prefixes that it can reach via the CE router it is connected to. The Multi-Protocol extension to BGP is needed to introduce the concept of the "customer" (i.e., the "L3VPN identifier") which does not exist in plain BGP.

Compared with any ad-hoc signalling mechanism that could have been designed specifically for MPLS, the choice of using BGP has the advantage of relying on a well-known protocol and thus making the learning curve smoother for practitioners. Moreover, BGP has built-in mechanisms (e.g., route reflection) to scale as the number of PE routers increases.

Fig. 13 shows a high-level illustration of how the BGP peerings with LER3 and LER2 can be used by LER1 to announce customer prefixes.

Configuring MP-BGP peerings is very similar to configuring plain iBGP peerings. Consider the following snippet from the configuration of router LER1:

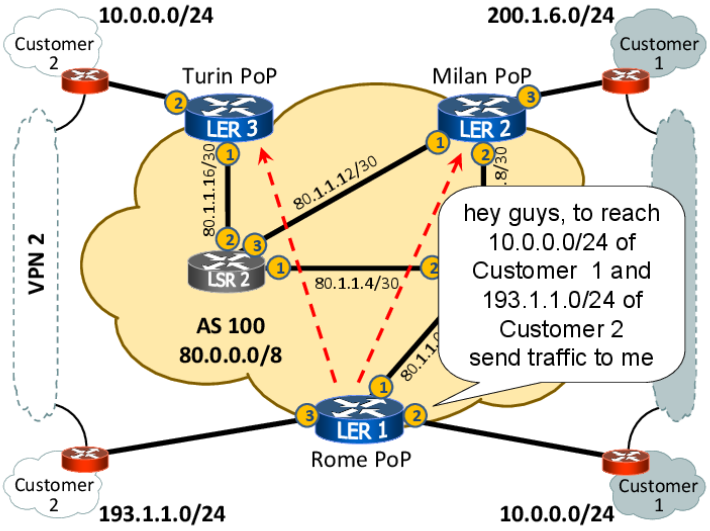

Figure 13: Use of BGP to distribute customer prefixes.

```
router bgp 100
  neighbor 80.80.80.4 remote-as 100
  neighbor 80.80.80.4 update-source Loopback0
  neighbor 80.80.80.5 remote-as 100
  neighbor 80.80.80.5 update-source Loopback0
  !
  address-family vpnv4
    neighbor 80.80.80.4 activate
    neighbor 80.80.80.5 activate
  exit-address-family
```

The first line starts the BGP configuration and states that the router belongs to AS100. Observe that all the routers are supposed to belong to Autonomous System (AS) 100. This AS number will not be necessarily propagated outside the provider's network and is only needed to establish peerings between PEs.

The following lines specify the BGP peerings. The presence of the "`vpnv4`" address family identifies LER2 and LER3 as MP-BGP neighbors of LER1.

3.3 Move 3 – Use MPLS encapsulation among PEs

Having performed Move 1 and Move 2, a PE router r is able to select the PE router r' that is connected to a given customer prefix (by Move 2). Also, r is able to forward IP packets to r' (by Move 1). The only piece missing is an encapsulation mechanism to transport IP packets from r to r'. One such encapsulation mechanism is MPLS: the PE router r encapsulates the IP packet by pushing two MPLS labels. The label at

the top of the stack (*outer* label) is switched by P routers in order to deliver the packet to router r'. The label at the bottom of the stack (*inner* label) is left untouched and it is used by the egress PE r' to identify the correct L3VPN. Observe that the inner label is necessary because r and r' could serve a variety of customers, and address spaces might be overlapping. For instance, routers r and r' could be serving two distinct VPNs for two customers, both using addresses in the RFC 1918 space.

Let us briefly recap how a packet is delivered across an MPLS cloud. When PE router r receives a packet from a CE router, it picks the VPN identifier and the destination address and, based on information contained in its MP-BGP Routing Information Base (RIB), it finds the PE router the packet should be delivered to. The MP-BGP RIB also contains the inner label that should be used. In our running example, the egress PE router is r'. Then, r pushes the inner MPLS label and an outer label which is guaranteed to deliver the packet to r'.

How does r pick this outer label? The outer label that maps to router r' is determined by the Label Forwarding Information Base (LFIB) of r, which is the forwarding table for MPLS.

The task of distributing labels and maintaining the LFIB of label switch routers is performed by the Label Distribution Protocol (LDP)[1][2]. LDP is able to setup a Label Switch Path (LSP) from one PE to another. In its simplest form, LDP maps an address prefix FEC (i.e., a FEC which represents an IP prefix) to a label. Each router receives LDP mappings from all its neighbors. In order to populate the LFIB, each router inspects its own forwarding table to determine the IP nexthop for the FEC, and picks the corresponding label. This way, LDP effectively creates a forwarding tree rooted at the egress point of the FEC, by simply importing the nexthop from the IP data plane (remember Move 1) at each intermediate hop.

It is extremely simple to configure a router to fulfill Move 3, because the LDP protocol can be safely run in the default configuration, and enabling MPLS encapsulation on specific interfaces is a single command. The configuration of LER1 for Move 3 is as simple as the following.

```
mpls label protocol ldp
interface GigabitEthernet1/0
  ip address 80.1.1.1 255.255.255.252
  mpls ip
```

4 An In-Depth View of MPLS VPNs

We have seen that the architecture of MPLS VPNs builds upon three building blocks: a working IP data plane that is capable of interconnecting the loopback addresses of PE routers, a BGP-based control plane to distribute reachability information about customer prefixes, and MPLS encapsulation among PEs.

While the first building blocks might seem straightforward at a first glance, there are a number of details which complicate the big picture but nevertheless are important in order to grasp the internals of MPLS VPNs.

4.1 IP Data Plane

IP connectivity between PE routers is easy to achieve using any suitable routing protocol. However, PE routers might be attached to a number of CEs of different customers, and must ensure that each CE is

[2] Alternative protocols such as RSVP and BGP can also serve the same purpose, but are out of the scope of this chapter.

VRF id	Ingress interface	Destination address	Egress interface
VRF-3	en5	10.100.200.32	en2

Table 4: Structure of the forwarding table in the "forwarding by network address" approach with Virtual Routing and Forwarding (VRF).

mapped to the correct VPN. A traditional IP data plane is unfit for this purpose since the IP address space of customers can overlap. Hence, a PE router must be able to route packets based on both the IP address and the specific VPN the packet belongs to. To accomplish this task, MPLS VPNs exploit a technique called Virtual Routing and Forwarding (VRF) which allows a router to have multiple (virtual) routing tables, potentially a separate virtual routing table for each network interface (either physical or logical). With this technique, mapping a CE to the correct VPN is as easy as configuring the corresponding interface within a specific VRF table. An MPLS inner label actually identifies a VRF instance.

One way to implement VRF while still maintaining a single forwarding table is using the ingress interface as an additional input parameter in the forwarding table. In such an implementation, the organization of the forwarding table of a router would be the one illustrated in Table 4. Observe that only the PE router needs to support VRF. The CE router is configured with a plain eBGP configuration and is completely unaware of the VRF implemented on the provider side.

Assigning an interface to a specific VRF instance is straightforward. In our sample network, we configure router LER1 as follows.

```
interface GigabitEthernet2/0
  ip vrf forwarding VPN1
  ip address 10.0.0.1 255.255.255.0
interface GigabitEthernet3/0
  ip vrf forwarding VPN2
  ip address 193.1.1.1 255.255.255.0
```

Address `10.0.0.1` is the address assigned to the interface that connects LER1 to Customer 1, while address `193.1.1.1` is assigned to the interface that connects LER1 to Customer 2.

4.2 MP-BGP Control Plane

Multi-protocol extensions to BGP allow us to segregate each L3VPN in a different *namespace*, identified by a proper VPN identifier. Separating namespaces is important because an IP prefix is not guaranteed to be unique across multiple VPNs: in practice, customers might want to use their own private IP address spaces, possibly overlapping with the address space of other customers. MP-BGP solves this issue by introducing the concept of VPN-IP addresses, that is, IP addresses tagged with a 8-byte VPN identifier which is called *route distinguisher* (RD). A VPN-IP address is nothing more than the concatenation of the RD and the IP prefix. By imposing that different VPNs be assigned distinct RD values, the uniqueness of VPN-IP addresses is guaranteed even in the presence of overlapping IP address space among customers. A special RD value consisting of 8 NULL bytes represent the default VPN, which allows MP-BGP to distribute information regarding pure IP routes alongside information about VPN-IP prefixes.

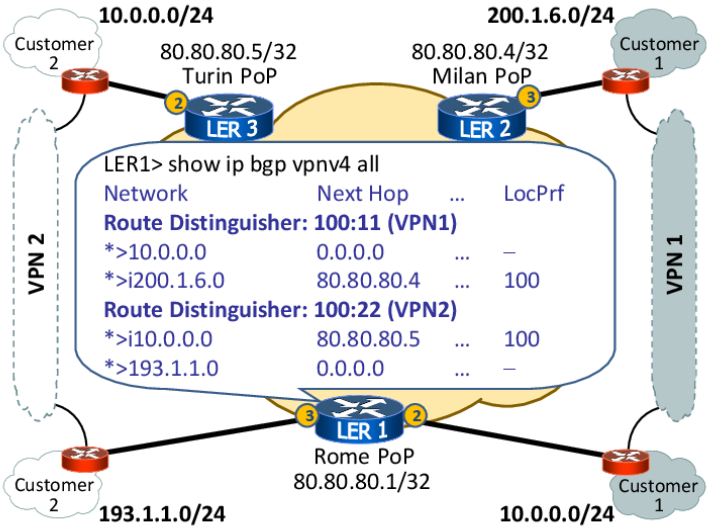

Figure 14: How MP-BGP can distribute per-VPN reachability information. Observe that the Local Preference attribute is not meaningful for locally originated prefixes, which are automatically preferred by the BGP decision process.

Observe that, while a VPN-IP prefix uniquely identifies a destination, it provides no information about the reachability of that destination. For this reason, MP-BGP messages associate VPN-IP prefixes with the MPLS labels that should be used for forwarding.

It is easy to assign an RD value to a single VRF instance:

```
ip vrf VPN1
  rd 100:11
ip vrf VPN2
  rd 100:22
```

Fig. 14 shows the output of command `show ip bgp vpnv4 all` on router LER1. This command has the same effect of `show ip bgp` but it shows the routing entries related to IPv4 VPNs. In this case the output highlights that LER1, in addition to its locally originated prefixes 10.0.0.0/24 with Route Distinguisher 100:1 and 193.1.1.0/24 with Route Distinguisher 100:2, knows two remote prefixes. Namely, it knows 200.1.6.0 with Route Distinguisher 100:11 and 193.1.1.0/24 with Route Distinguisher 100:22.

Tagging IP prefixes with a VPN identifier is an easy solution, but it is suboptimal in a specific use case which has seen increasing popularity recently: the so-called *extranets*. In its simple definition, an extranet is simply a connection between two different VPNs that are guaranteed to have non-overlapping IP address spaces. A realistic example might be a specific site of one customer that needs to connect to another specific site of another customer. A naive implementation of extranets would define an ad-hoc VPN and assign it a

new RD value. However, this solution is undesirable because it creates multiple VPNs that have duplicate entries, yielding a waste of router memory (to store the entries) and a waste of router's CPU time (to process update messages that are identical but for the RD value).

In order to overcome such limitations, MPLS decouples the concept of route distinguisher, which is used to segregate the address space in multiple namespaces, from the concept of *route target* (RT) which is another tag that is used to control which routes are imported in a given VPN and, similarly, which routes are exported from a given VPN. The route target is transported by MP-BGP using extended communities. More precisely, by exporting a route from a VPN we attach a user-defined RT community to all VPN-IP prefixes belonging to that VPN. On the other hand, by importing a given RT into a VPN we accept that every route having that RT value will be visible from the devices in that VPN.

Each VRF instance can be configured to import or export routes labelled with a specific Route Target value. In our simple example, assuming that no extranet connectivity is required between Customer 1 and Customer 2, each VRF instance can simply import a single RT value, as the following configuration snippet of LER1 shows:

```
ip vrf VPN1
  rd 100:11
  route-target export 100:1000
  route-target import 100:1000
ip vrf VPN2
  rd 100:22
  route-target export 100:2000
  route-target import 100:2000
```

This means that all the prefixes of VPN1 announced via MP-BGP by LER1 to any other PE are tagged with RT 100:1000. Also, any prefix that is tagged 100:1000 and is announced to LER1 via MP-BGP is imported into the VRF of VPN1. The configuration for VPN2 is similar.

Route Targets provide network operators with the flexibility of leaking specific routes into specific VRF instances, easing the deployment of extranets. Route Targets are transported in MP-BGP messages as extended BGP communities. For this reason, the configuration of MP-BGP peers needs to specify that the peer supports extended communities (which are disabled by default).

```
router bgp 100
  address-family vpnv4
    neighbor 80.80.80.4 activate
    neighbor 80.80.80.4 send-community both
    neighbor 80.80.80.5 activate
    neighbor 80.80.80.5 send-community both
  exit-address-family
```

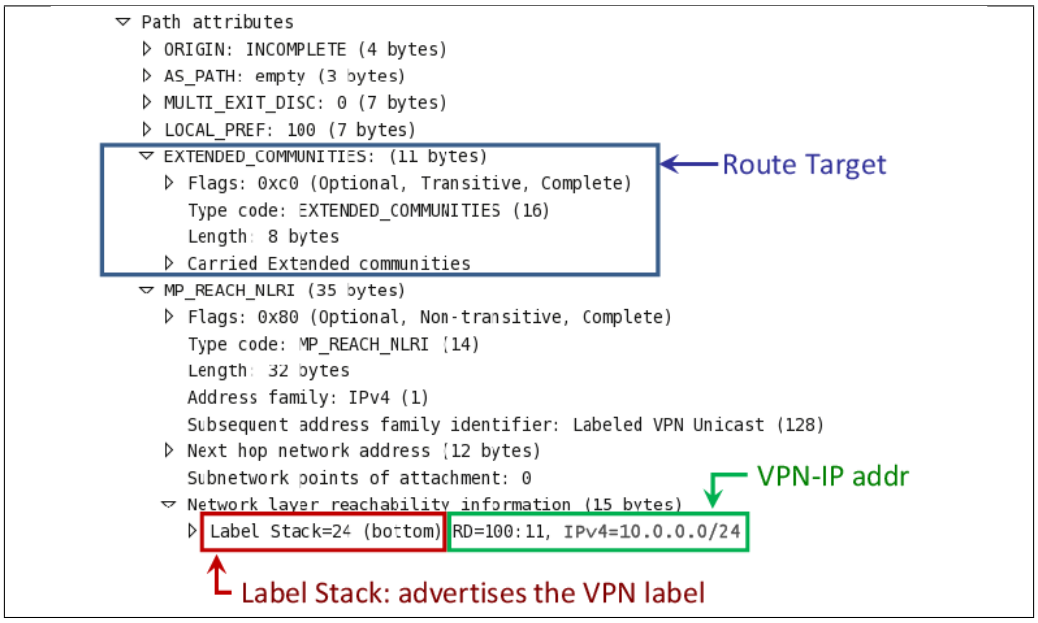

Figure 15: An MP-BGP signaling packet captured over the network.

To better understand the interplay between MP-BGP and the Route Targets, let us look at the content of an MP-BGP packet captured in our network (see Fig. 15). Observe how the route target in the blue frame is contained in the extended communities.

The announcements tells to the MP-BGP peer receiving it that the packets that will be received with the inner MPLS label 24 (red frame in the picture) will refer to the specified route target and the specified route distinguisher (green frame in the picture).

4.3 MPLS Control and Data Plane

The task of MPLS control plane is simply to establish Label Switched Paths (LSPs) between the loopback addresses of PE routers. LDP is in charge of populating and maintaining routers' LFIBs that implement the LSPs. In its most popular distribution mode, called *unsolicited downstream*, LDP works in the following way. Each router creates a label for locally originated prefixes (e.g., the loopback address). The binding between a label and a locally originated prefix is called a *local* binding. By contrast, a *remote* binding is a binding between a label and a remotely originated prefix. Each router advertises its local bindings to its neighbors. When a neighboring router receives a binding for prefix p_1 and label l_1 on interface i_1, it looks up its IP forwarding table to check whether the advertised prefix is routed on interface i_1. If this is the case, it picks another label l_2 and starts announcing a binding for p_1 and l_2. Meanwhile, it updates its LFIB with the tuple $\langle l_2, l_1, p_1, i_1 \rangle$. This means that when a packet arrives that is labelled l_2, the LFIB will swap l_2 with

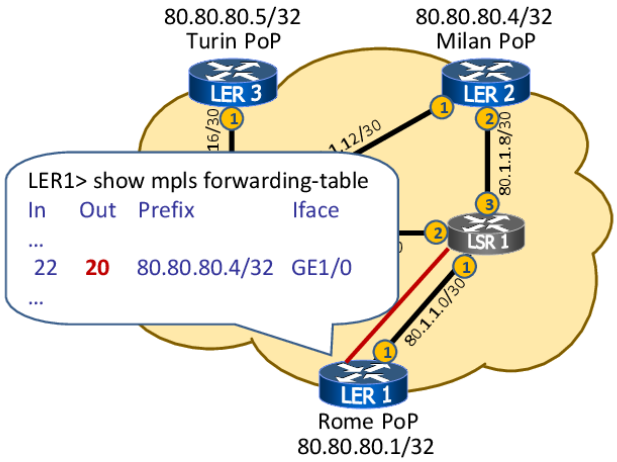

Figure 16: The MPLS forwarding table of LER1.

l_1 and deliver it via interface i_1[3].

Regarding MPLS data plane, we have already seen that the ingress PE router looks up its MP-BGP RIB to find the loopback address of the egress PE router, looks up its LFIB to select the outer label, and then encapsulates the received IP packet by pushing the inner and the outer MPLS labels. The packet is then label-switched across the MPLS network to the egress PE router using the bindings found in the LFIB of each router. As an optimization, the penultimate router, i.e., the router that receives a local binding from the egress PE, can pop the outer label, in such a way that the egress PE router only receives the inner label and therefore performs a single lookup in its LFIB.

Figs. 16, 17, and 18 show the MPLS forwarding tables of some routers of our network.

Figs. 19–24 illustrate the travel of a packet through our network.

First, Fig. 19 shows what happens when an IP packet originated by the Rome site of Customer 1 reaches the PE called LER1. Namely, it is encapsulated into an MPLS packet with two labels and then sent to LSR1. The inner label (yellow) identifies the VRF while the outer label (red) is the label used for the forwarding process.

Second (Fig. 20), the MPLS packet reaches LSR1, its outer red label is replaced with a blue label, and the packet is forwarded to LSR2.

Third (Figs. 21 and 22), the MPLS packet reaches LSR2. LDP makes LSR2 aware that it is the penultimate hop in the LSP. For this reason, LSR2 simply pops the outer label and forwards the packet to LER2.

[3]This explanation assumes per-router label scope, which is the default for most router vendors. An alternative is per-interface label scope, when the router can advertise different labels for each interface, which is used mostly on ATM interfaces.

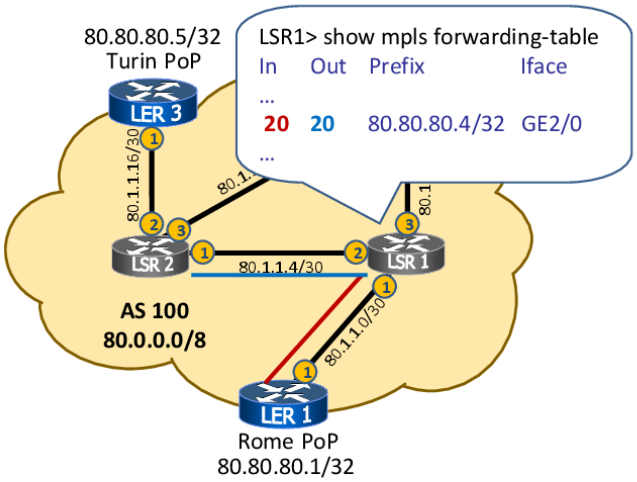

Figure 17: The MPLS forwarding table of LSR1.

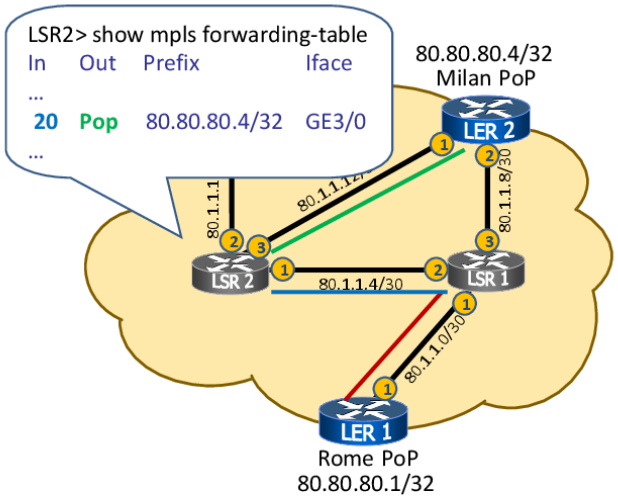

Figure 18: The MPLS forwarding table of LSR2.

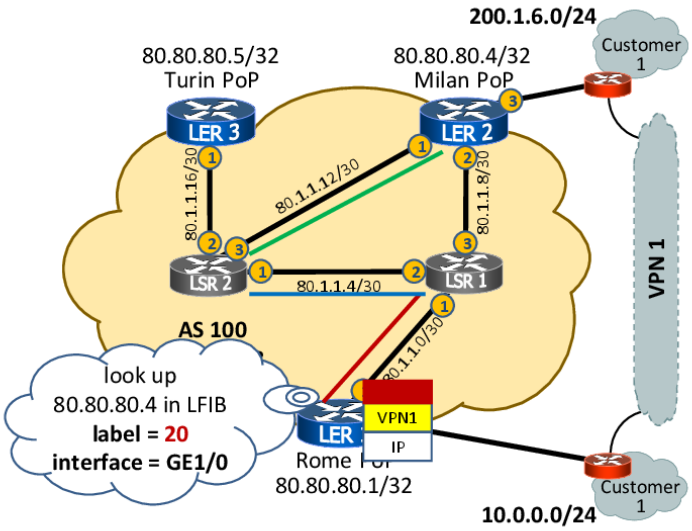

Figure 19: An IP packet originated by the Rome site of Customer 1 reaches PE router LER1.

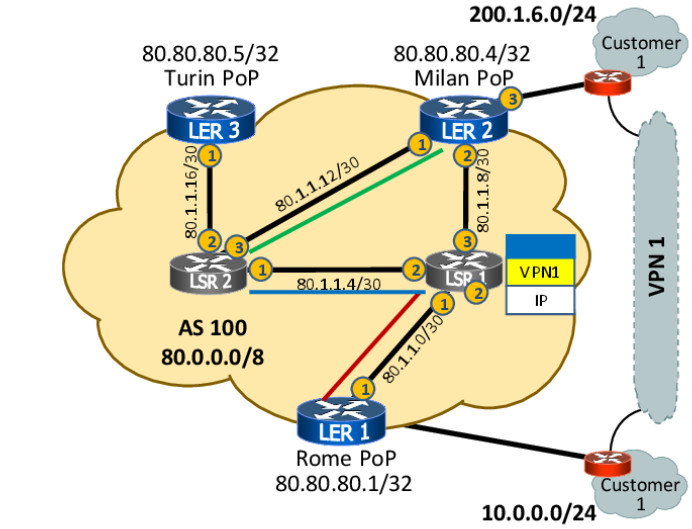

Figure 20: An MPLS packet reaches P router LSR1.

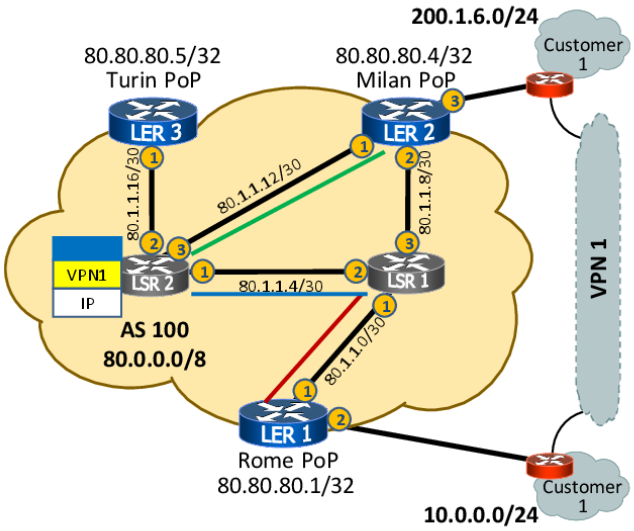

Figure 21: An MPLS packet reaches P router LSR2.

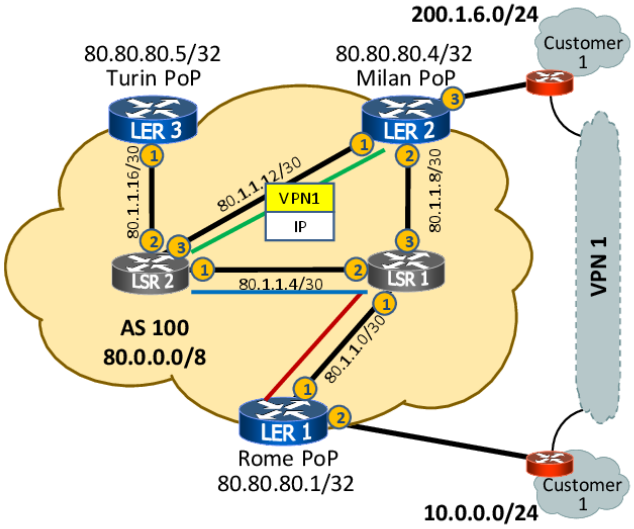

Figure 22: An MPLS packet traveling to PE router LER2.

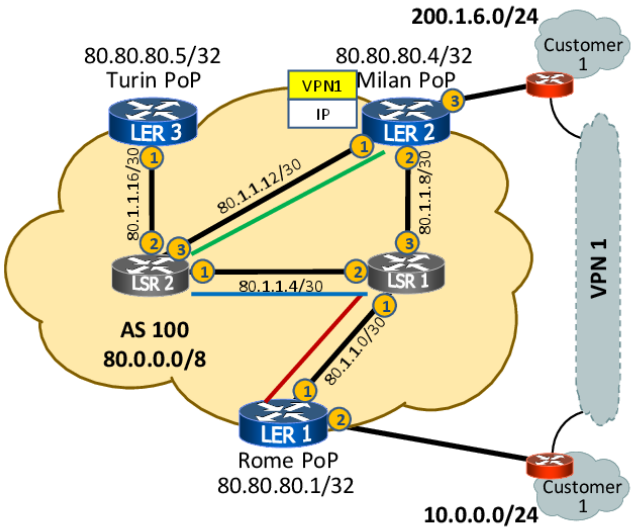

Figure 23: An MPLS packet reaches PE router LER2.

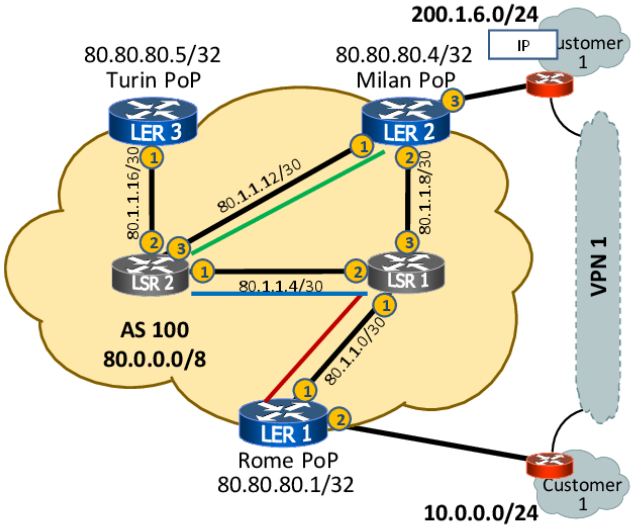

Figure 24: An IP packet for Customer 1.

Fourth (Fig. 23), the MPLS packet reaches LER2. LER2 notices that there is only one MPLS label (which used to be the inner label), so the packet is meant to be forwarded via IP in one of the VPNs that LER2 serves. LER2 uses the VRF label to identify the VRF instance, then looks up the IP destination address in the VRF forwarding table.

Fifth (Fig. 24), the IP packet is delivered to its final destination.

5 Advanced Topics

In this section we give more technical details about dynamic routing and connecting to the Internet, creating complex VPN topologies, and dealing with IP-specific features (e.g., MTU) that need extra care when encapsulation is involved.

5.1 Dynamic Routing and Connecting to the Internet

So far we have not yet discussed how the PE router can learn the prefixes that are served by its directly attached CE router. Of course, it is trivial to configure static routes on the PE, however this creates an undesirable coupling between the provider and the customer: whenever the customer wants to add a different IP subnet, it has to bother the provider to configure static routes before that IP subnet is reachable from other customer sites in the same VPN.

The solution is to have the CE and the PE establish an eBGP peering where the CE announces its local networks, while the PE announces all the networks that it learns in the same VPN. Observe that the, contrary to the MP-BGP peerings among PEs, peerings between CEs and PEs are pure eBGP peering: the CE does not know anything about VPNs and route distinguishers. It is the MP-BGP process on the PE router that takes care of processing the reachability information learned from the CE and updating the VPN reachability information accordingly.

Setting up an eBGP session in order to use the MPLS-VPN service may discourage those customers who do not have a strong BGP expertise. In such cases, usually the providers also offer to their customers CE management and configuration.

A BGP peering between the CE and the PE also allows a CE in a VPN to announce a default route, causing all other sites in the same VPN to route Internet traffic via that CE router. This might be advantageous if the customer's policy forces Internet traffic to pass through a centralized checkpoint (e.g., a firewall or a proxy). However, this is not the only way to connect a VPN to the Internet. For example, a PE might be configured to forward natively all the packets from a CE which do not match any VPN route. Alternatively, the default route might be given its own route target, and whenever a VPN site needs Internet access the PE simply imports that route target in the corresponding VPN routing table. We refer the reader to [14] for a discussion of alternatives to get Internet access within a VPN.

In our sample network, customer 1 would like to be able to create a new IP subnet in Rome, advertise the new subnet to LER1 via an eBGP peering, and automatically make the customer site in Milan able to access it. LER1 should receive the new route via eBGP, tag the route with the correct RD value, and advertise it in MP-BGP. In order to do this, it suffices to configure an eBGP peering in the context of a VRF instance, as the following configuration snippet of LER1 shows.

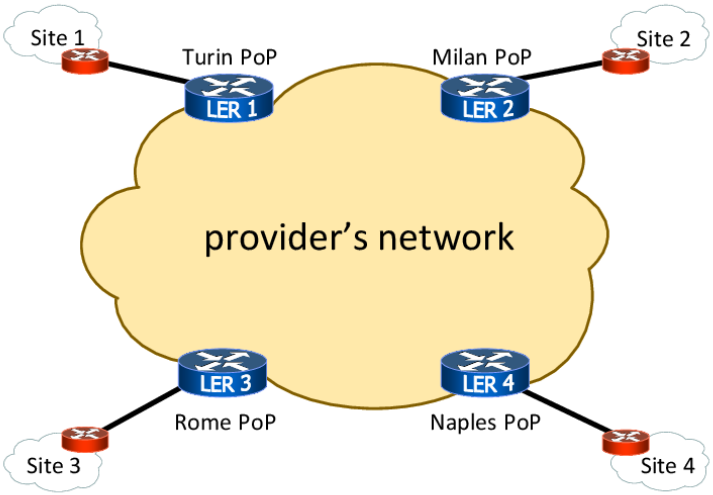

Figure 25: A configuration where a single customer has four sites: Site 2, 3, and 4 are only allowed to exchange traffic with Site 1 in Rome.

```
router bgp 100
  address-family ipv4 vrf VPN1
  neighbor 10.0.0.2 remote-as 65001
  exit-address-family
  !
  address-family ipv4 vrf VPN2
  neighbor 193.1.1.2 remote-as 65002
  exit-address-family
```

5.2 Designing Complex VPNs

So far we have assumed any-to-any connectivity within a VPN, i.e., all sites of a VPN communicate with all other sites. Sometimes more sophisticated configurations are needed. For example we might have a VPN where not all pairs of sites are allowed to exchange packets. A typical situation is the so called hub-and-spoke configuration, where a customer has a main site and several peripheral sites and the peripheral sites can communicate only through the main site.

How to do this is illustrated in the following example.

A suitable use of Route Distinguishers and Route Targets allows sophisticated configurations like the one shown in Fig. 25 where Rome is the main site and Turin, Milan, and Naples are the peripheral sites.

We can choose Route Distinguisher `100:1` for all four sites and split the customer VPN into three VPNs. VPN1 is used to connect Turin with Rome, VPN2 is used to connect Milan with Rome, and VPN3 is used to connect Naples with Rome.

For each VPN we define a distinct Route Target:

VPN1: `100:1000`

VPN2: `100:2000`

VPN3: `100:3000`

The configuration of peripheral sites, like for example Turin, is as follows:

```
ip vrf siteTurin
        rd 100:1
        route-target import 100:1000
        route target export 100:1000
```

Rome's PE configuration (the hub) is as follows:

```
ip vrf siteRome
        rd 100:1
        route-target import 100:1000
        route target export 100:1000
        route-target import 100:2000
        route target export 100:2000
        route-target import 100:3000
        route target export 100:3000
```

In this way Rome imports all the Route Targets and exports all the Route Targets and is hence able to communicate with all sites. On the other hand a peripheral site like Turin imports and exports Route targets only with respect to Rome and hence is able to communicate with Rome only.

5.3 ToS, TTL, and MTU

Whenever encapsulation of IP packets happens, there are three main questions that arise:

1. what happens to the ToS / DSCP information in the IP header that the customer might have set in order to properly prioritize traffic?
2. what happens to the TTL field in the IP header and how does encapsulation cope with forwarding loops?
3. how does encapsulation affect MTU for upper layer protocols?

Luckily, MPLS has an easy answer for the first two questions. Recall from Fig. 6 that MPLS has dedicated fields for ToS and TTL. When the ingress PE router receives an IP packet from the CE router, it simply

copies the values of ToS and of TTL in the MPLS header. More precisely, a push operation implies copying ToS and TTL from the IP header to the MPLS header. Conversely, a pop operation implies copying the TTL value from the MPLS header back to the IP header. This way, the TTL continues to serve as a hop count[4] even within the MPLS network, and P routers can honor the quality of service parameters related to the ToS field.

Regarding the third question, since an MPLS label takes 4 bytes and the PE router pushes two of them, the MTU within the MPLS network should be at least 8 bytes larger than the MTU that the CE is aware of. Given that modern OSes tend to perform path MTU discovery by default, MTU is becoming less of an issue for MPLS deployments. Rewriting the Maximum Segment Size TCP options at the PE router is also a common solution, even though it does not support UDP traffic.

6 Summary

6.1 Strengths of MPLS VPNs

After having described the details of MPLS VPNs, we are able to discuss the extent to which the goals that we stated in Section 1.2 are met.

By using Route Distinguishers and label stacks within the provider cloud, customers can retain their IP address plan and the traffic belonging to different customers is properly segregated. Moreover, the configuration of CE routers is completely unaware of MPLS-specific details. Since MPLS transports QoS information by copying the ToS field from the IP header, MPLS VPNs can in principle provide different forwarding treatment to different packets. However, the architecture of MPLS VPNs does not inherently support QoS, because LSPs are simply built from the underlying IP plane. Other mechanisms (e.g., [4]) can be employed to compute LSPs based on QoS features.

Providers are able to keep the configuration in the core of the network extremely simple and scalable: in fact, the configuration of P routers does not depend on the number of deployed VPNs. Since the backbone is only concerned with transporting packets from a PE to another, the sizeof the forwarding table of P routers only depends on the number of PEs and does not depend on the number of prefixes of VPNs. Configuring a new VPN implies modifying the configuration of the PE routers that are directly connected to the customer's sites. Moreover, such a configuration boils down to assigning a unique RD and RT and establishing eBGP peerings with the CE routers.

6.2 Limitations of MPLS VPNs

Virtual Private Networks designed with MPLS have also some known limitations. At least the following should be mentioned.

- BGP know-how is needed to configure the customer CE. As discussed in Section 5.1, this sometimes forces the carrier to provide also CE management.
- Customer CEs need to support BGP if eBGP peering are established with the PEs. This may not be the case for cheap entry-level router models. As an alternative, BGP can be replaced by static routes

[4]Observe that when the TTL in the MPLS header reaches 0 (e.g. in a traceroute), a P router does not know how to send the corresponding ICMP error back to the sender, because it lacks information about VPNs. A naive yet effective solution is to generate the ICMP packet and label-switch it to the egress PE anyway. The egress PE (which has information about VPNs) will then send the ICMP packet back to the sender.

configured on PEs, sacrificing part of the flexibility that a dynamic routing protocol between CEs and PEs provides.

- The P in the MPLS acronym stands for "Private". However, MPLS VPNs are only private at routing level: no authentication, confidentiality, or integrity is provided by the architecture. For instance, the provider can inspect all customers' traffic in plaintext. Even worse, since the separation is enforced at the routing level, it turns out that the ability of guaranteeing isolation within the same VPN actually depends on the provider's topology [6].

- The basic MPLS architecture lacks support for quality of service. QoS can usually be offered on top of MPLS, e.g., by establishing LSPs which reflect traffic engineering policies. However, adding traffic engineering and quality of service support comes at the cost of keeping more state in the network, hence posing scalability concerns [12, 19].

- In most configurations, the customer and the provider share the job of maintaining a network, which potentially complicates debugging routing and connectivity problems.

6.3 Further Readings

The interested reader could refer to the classical books about MPLS authored by Minei and Lucek [11] and by De Ghein [9].

Most of the technologies regarding MPLS are defined by RFCs. These include the main MPLS VPNs architecture [15, 7], label distribution via LDP [1], Layer 2 VPNs [2], BGP variants [5, 14, 13], and RSVP-TE [4].

Considerations about about MPLS VPNs integrity and scalability can be found in [6] and [12, 19], respectively.

Acknowledgments

We thank the anonymous reviewers for constructive criticism and for suggestions that helped us improve both the content and the presentation of this chapter. We also thank Mario Cola and Massimo Rimondini for their help and friendship.

References

[1] Andersson, L., Minei, I., and Thomas, B. LDP Specification. RFC 5036, 2007.

[2] Andersson, L., and Rosen, E. Framework for Layer 2 Virtual Private Networks (L2VPNs). RFC 4664, 2006.

[3] Augustyn, W., and Serbest, Y. Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks. RFC 4665, 2006.

[4] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and Swallow, G. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, Dec. 2001.

[5] Bates, T., Chandra, R., Katz, D., and Rekhter, Y. Multiprotocol Extensions for BGP-4. RFC 4760, 2007.

[6] BUSH, R., AND GRIFFIN, T. G. Integrity for virtual private routed networks. In *In Proc. IEEE INFOCOM* (2003).

[7] CALLON, R., AND SUZUKI, M. A Framework for Layer 3 Provider-Provisioned Virtual Private Networks. RFC 4110, 2005.

[8] CARUGI, M., AND MCDYSAN, D. Service Requirements for Layer 3 Provider-Provisioned Virtual Private Networks. RFC 4031, 2005.

[9] DE GHEIN, L. *MPLS Fundamentals*. Cisco Press, Dec. 2006.

[10] FULLER, V., AND LI, T. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632, 2006.

[11] MINEI, I., AND LUCEK, J. *MPLS-Enabled Applications: Emerging Developments and New Technologies*. Wiley, Oct. 2005.

[12] MINEY, I. Scaling considerations in MPLS networks. Nanog 35, 2005.

[13] MOHAPATRA, P., AND ROSEN, E. The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute. RFC 5512, 2009.

[14] ROSEN, E., AND REKHTER, Y. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, 2006.

[15] ROSEN, E., VISWANATHAN, A., AND CALLON, R. Multiprotocol Label Switching Architecture. RFC 3031, 2001.

[16] T. WORSTER, Y. R., AND ROSEN, E. Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE). RFC 4023, 2005.

[17] TOWNSLEY, M. MPLS over various IP tunnels. Nanog 30, 2004.

[18] VARGHESE, G. *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann, Dec. 2004.

[19] YASUKAWA, S., FARREL, A., AND KOMOLAFE, O. An Analysis of Scaling Issues in MPLS-TE Core Networks. RFC 5439, Feb. 2009.