Revealing and Characterizing MPLS Networks

YVES VANAUBEL

Department of Electrical Engineering and Computer Science
Faculty of Applied Science
UNIVERSITY OF LIÈGE - BELGIUM

SUPERVISOR: PROF. BENOIT DONNET

This dissertation is submitted for the degree of
DOCTOR IN APPLIED SCIENCE

OCTOBER 2018

Montefiore Institute

ABSTRACT

The Internet is a wide network of computers in constant evolution. Each year, more and more organizations are connected to this worldwide network. Each of them has its own structure and administration that are not publicly revealed for economical, political, and security reasons. Consequently, our perception of the Internet structure, and more specifically, its topology, is incomplete. In order to balance this lack of knowledge, the research community relies on network measurements. Most of the time, they are performed based on the well-known tool `traceroute`. However, in practice, an operator may privilege other technologies than IP to forward packets inside its network. *MultiProtocol Label Switching (MPLS)* is one them. Even if it is heavily deployed by operators, it has not been really investigated by researchers. Prior to this thesis, only two studies focused on the identification of MPLS tunnels in `traceroute` data. Moreover, while one of them does not take all possible scenarios into account, the other lack of precision in some of its models. In addition, MPLS tunnels may hide their content to `traceroute`. Topologies inferred from such data may thus contain false links or nodes with an artificially high degree, leading so to biases in standard graph metrics used to model the network. Even if some researchers already tried to tackle this issue, the revelation of hidden MPLS devices in `traceroute` data is still an open question.

This thesis aims at characterizing MPLS in two different ways. On the one hand, at an architectural level, we will analyze in detail its deployment and use in both IPv4 and IPv6 networks in order to improve its state-of-the-art view. We will show that, in practice, more than one IPv4 trace out of two crosses at least one MPLS tunnel. We will also see that, even if this protocol can simplify the internal architecture of transit networks, it also allows some operators to perform traffic engineering in their domain. On the other hand, MPLS will be studied from a measurement point of view. We will see that routers from different manufacturers may have distinct default behaviors regarding to MPLS, and that these specific behaviors can be exploited to identify MPLS tunnels during `traceroute` measurements. More precisely, we will focus on new methods able to infer the presence of tunnels that are invisible in `traceroute` outputs, as well as on mechanisms to reveal their content. We will also show that they can be used in order to improve the inference of Internet graph properties, such as path lengths and node degrees. Finally, these techniques will be integrated into *Trace the Naughty Tunnels (TNT)*, a `traceroute` extension able to identify all types of MPLS tunnels along a path towards a destination. We will prove that this tool can be used in order to get a detailed quantification of MPLS tunnels in the worldwide network. `TNT` is publicly available, and can therefore be part of many future studies conducted by the research community.

Keywords: MPLS; traceroute; topology; network discovery; IPv6; 6PE; traffic engineering; fingerprinting

RÉSUMÉ

Internet est un immense réseau informatique en constante évolution. Chaque année, de plus en plus d'organisations s'y connectent. Chacune d'elles est gérée et administrée indépendamment des autres. En pratique, l'architecture interne de leur réseau n'est pas rendue publique pour des raisons politiques, économiques, ou de sécurité. Par conséquent, notre perception de la structure d'Internet, et plus particulièrement de sa topologie, est incomplète. Afin de pallier ce manque de connaissance, la communauté de la recherche s'appuie sur des mesures de réseau. La plupart du temps, elles sont réalisées avec l'outil `traceroute`. Cependant, des technologies autres que IP peuvent être privilégiées pour transférer les paquets dans un réseau. *MultiProtocol Label Switching (MPLS)* est l'une d'entre elles. Même si cette technologie est largement déployée dans Internet, elle n'est pas bien étudiée par les chercheurs. Avant cette thèse, seulement deux travaux se sont intéressés à l'identification d'MPLS dans les données collectées avec `traceroute`. Alors que le premier ne prend pas en compte tous les scénarios possibles, le second propose des modèles qui manquent de précision. De plus, les tunnels MPLS peuvent dissimuler leur contenu à `traceroute`. Les topologies inférées sur base de ces données peuvent donc contenir de faux liens, ou des noeuds avec un degré anormalement élevé. Les différentes modélisations d'Internet qui en résultent peuvent alors être biaisées. Aujourd'hui, la question de la révélation des routeurs MPLS qui sont invisibles dans les données de mesure n'est toujours pas résolue, même si certains chercheurs ont déjà proposé quelques méthodes pour y parvenir.

Cette thèse a pour but de caractériser MPLS de deux manières différentes. Dans un premier temps, au niveau architectural, nous analyserons en détail son déploiement et son utilisation dans les réseaux IPv4 et IPv6 afin d'améliorer l'état de l'art. Nous montrerons qu'en pratique, plus d'une trace IPv4 sur deux traverse au moins un tunnel MPLS. Nous découvrirons également que bien que ce protocole peut être utilisé pour simplifier l'architecture interne des réseaux de transit, il peut aussi être déployé pour la mise en place de solutions d'ingénierie de trafic. Dans un second temps, MPLS sera étudié d'un point de vue mesure. Nous verrons que les comportements par défaut liés au protocole varient d'un fabricant de routeur à l'autre, et qu'ils peuvent être exploités afin d'identifier les tunnels MPLS dans les données `traceroute`. Plus précisément, nous découvrirons de nouvelles méthodes capables d'inférer la présence de tunnels invisibles avec `traceroute`, ainsi que de nouvelles techniques pour révéler leur contenu. Nous montrerons également qu'elles peuvent être utilisées afin d'améliorer la modélisation d'Internet. Pour terminer, ces techniques seront intégrées à *Trace the Naughty Tunnels (TNT)*, une extension de `traceroute` qui permet d'identifier tous les types de tunnels MPLS le long du chemin vers une destination. Nous prouverons que cet outil peut être utilisé pour obtenir des statistiques détaillées sur le déploiement d'MPLS sur Internet. `TNT` est disponible publiquement, et peut donc être librement exploité par la communauté de la recherche pour de multiples futures études.

Mots Clés: MPLS; traceroute; topologie; découverte de réseau; IPv6; 6PE; ingénierie de trafic; fingerprinting

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. B. Donnet. It has been a great pleasure to be his first Ph.D. student. I really appreciated his endless support and guidance during my years of research. His contribution and ideas made my Ph.D. experience exciting and rewarding.

I would like also to thank sincerely Profs. P. Mérindol and J.-J. Pansiot. Their help and advices were really valuable during the many studies we did together. They spent a lot of time to discuss with me the solutions to complex issues that I had to face during this thesis. I'm also very grateful for their warm welcomes during my different stays in Strasbourg.

I'm also really thankful to CAIDA members for their hospitality and support during my internship in San Diego. In particular, I would like to thank Prof. K. Claffy who offered me such a great opportunity to join her research team. I also really appreciated working with Y. Hyun to deploy `TNT` on the Archipelago Platform. His tremendous experience in Internet measurements was really helpful to me.

Many thanks to J.-R. Luttringer for the GNS3 validation of the different inference and revelation techniques implemented in `TNT`.

I'm also very grateful to my family and friends. I would especially like to thank my parents for their encouragement, and my partner, A.-C. Saal, for her warm love, patience, and support that were really precious to me during the writing of this dissertation.

Finally, I thank all those who contributed directly or indirectly to the achievement of this thesis.

AUTHOR'S DECLARATION

This thesis was written using the *University of Bristol Thesis Template* [24] created by V. BREÑA-MEDINA, and under the license *Creative Commons CC BY 4.0* [41]. The original template was modified in order to include parts and acronyms in the document.

Table of Contents

List of Tables

List of Figures

LISTINGS

ACRONYMS

ARIN American Registry for Internet Numbers.

ARPANET Advanced Research Projects Agency Network.

AS Autonomous System.

BGP Border Gateway Protocol.

BRPR Backward-Recursive Path Revelation.

CE Customer Edge.

DPR Direct Path Revelation.

ECMP Equal-Cost Multi-Path.

EH Exit Hop.

FEC Forwarding Equivalence Class.

FH First Hop.

FIB Forwarding Information Base.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

FTL Forward Tunnel Length.

HDN High-Degree Node.

IANA Internet Assigned Numbers Authority.

ICMP Internet Control Message Protocol.

IGMP Internet Group Management Protocol.

IGP Interior Gateway Protocol.

IMP Interface Message Processor.

IOTP In-Out Transit Pair.

IP Internet Protocol.

IS-IS Intermediate System to Intermediate System.

ISIS-TE Intermediate System to Intermediate System - Traffic Engineering.

ISP Internet Service Provider.

LDP Label Distribution Protocol.

LER Label Edge Router.

LFIB Label Forwarding Information Base.

LH Last Hop.

LIB Label Information Base.

LPR Label Pattern Recognition.

LSE Label Stack Entry.

LSP Label Switched Path.

LSR Label Switching Router.

MDA Multipath Detection Algorithm.

MP-BGP MultiProtocol - Border Gateway Protocol.

MPLS MultiProtocol Label Switching.

NPR No Path Revelation.

NSF National Science Foundation.

OS Operating System.

OSPF Open Shortest Path First.

OSPF-TE Open Shortest Path First - Traffic Engineering.

P Provider.

PE Provider Edge.

PH Penultimate Hop.

PHP Penultimate Hop Popping.

PR Path Revelation.

q-TTL Quoted TTL.

QoS Quality of Service.

RFPLA Return/Forward Path Length Asymmetry.

RIB Routing Information Base.

RIP Routing Information Protocol.

ROC Receiver Operating Characteristic.

RSVP Resource ReSerVation Protocol.

RSVP-TE Resource ReSerVation Protocol - Traffic Engineering.

RTL Return Tunnel Length.

SNMP Simple Network Management Protocol.

TC Traffic Class.

TCP Transmission Control Protocol.

TE Traffic Engineering.

TN True Negative.

TNT Trace the Naughty Tunnels.

TP True Positive.

TPR True Positive Rate.

TTL Time To Live.

UDP User Datagram Protocol.

UHP Ultimate Hop Popping.

VC Virtual Circuit.

VP Vantage Point.

VPN Virtual Private Network.

INTRODUCTION

Today's Internet is a network of computers that allows remote devices to exchange information. During a communication, the data to transfer is first divided into small parts, and stored inside packets. These packets are then transmitted one by one from the source to the destination, thanks to interconnected core elements called routers. Such a network is called packet-switched network.

Historically, the first exchange of data over a packet-switched network was performed in 1969. At that time, the precursor of the Internet, the *Advanced Research Projects Agency Network (ARPANET)*, allowed four remote US universities to communicate. This simple research network was composed of four interconnected *Interface Message Processors (IMPs)*, the first generation of routers, to which four computer devices were attached, as illustrated in Figure 1. It was used to develop and test multiple network protocols, such as the *Internet Protocol (IP)* [121] and the *Transmission Control Protocol (TCP)* [122]. ARPANET was funded by the US Government for research and development, and was not aimed to operate as an open communication network. In 1986, the *National Science Foundation (NSF)* created NSFNET [112], a network initially intended to share supercomputers between scientists and engineers around the country. It was based on the same technologies as ARPANET, and allowed institutions to exchange their knowledge at no cost. With the years, organizations started to connect to the existing architecture, and more and more elements were added to the original topology. This network became the first Internet backbone, shown in Figure 2. During the 90s, the first commercial companies were connected to it, and the privatization of the Internet started. In 1993, more than 2 million of computers were exchanging information over the global network [112]. Since that time, the Internet continued growing, and its internal structure became much more complex. In 2017, more that 3.5 billions users were interconnected [125]. A visualization of the core IPv4 Internet topology for the same year is available in Figure 3. It illustrates the interconnection of the different *Autonomous Systems (ASs)*[1] in the worldwide network.

Figure 3 shows that today's Internet is an ocean of interconnected networks. Each of them is owned by an organization that has its own structure and administration. Most of the time, operators do not reveal publicly the internal architecture of their network. In this way, they avoid competing companies copying their structure, or hackers identifying weaknesses that could be

[1] As defined in RFC1930 [77], "an AS is a connected group of one or more IP prefixes run by one or more network operators which has a single and clearly defined routing policy". An organization may own several ASs.

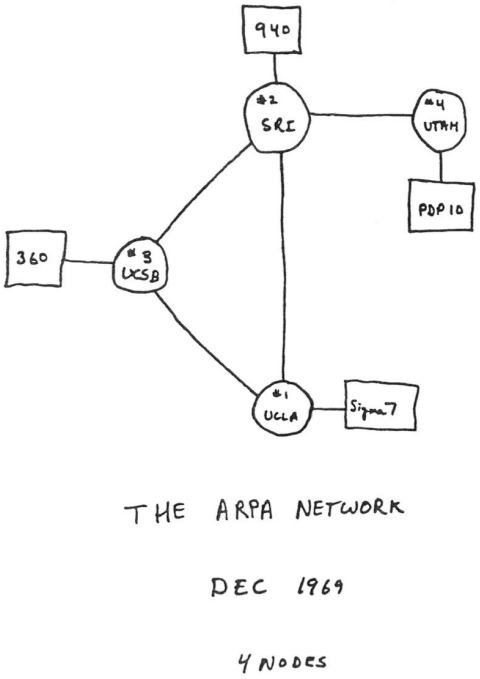


Figure 1: Topology of ARPANET, the first packet-switched network, in 1969 [40]. The first generation or routers (circles) interconnected four computers (rectangles) located in different universities in the USA.

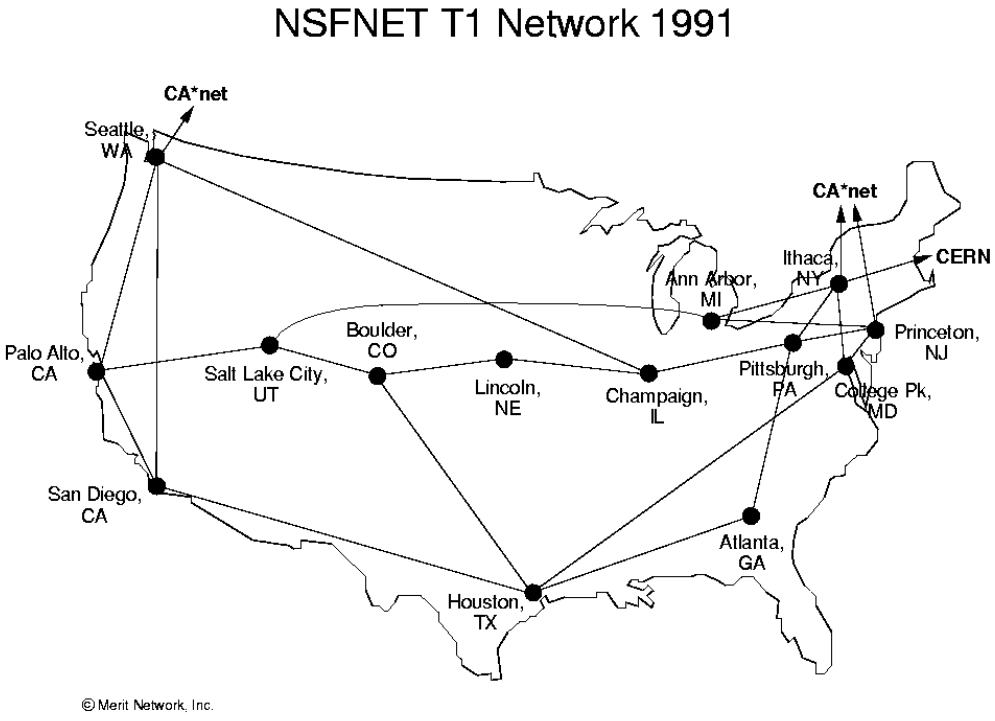


Figure 2: Topology of the NSFNET backbone in 1991 [108].

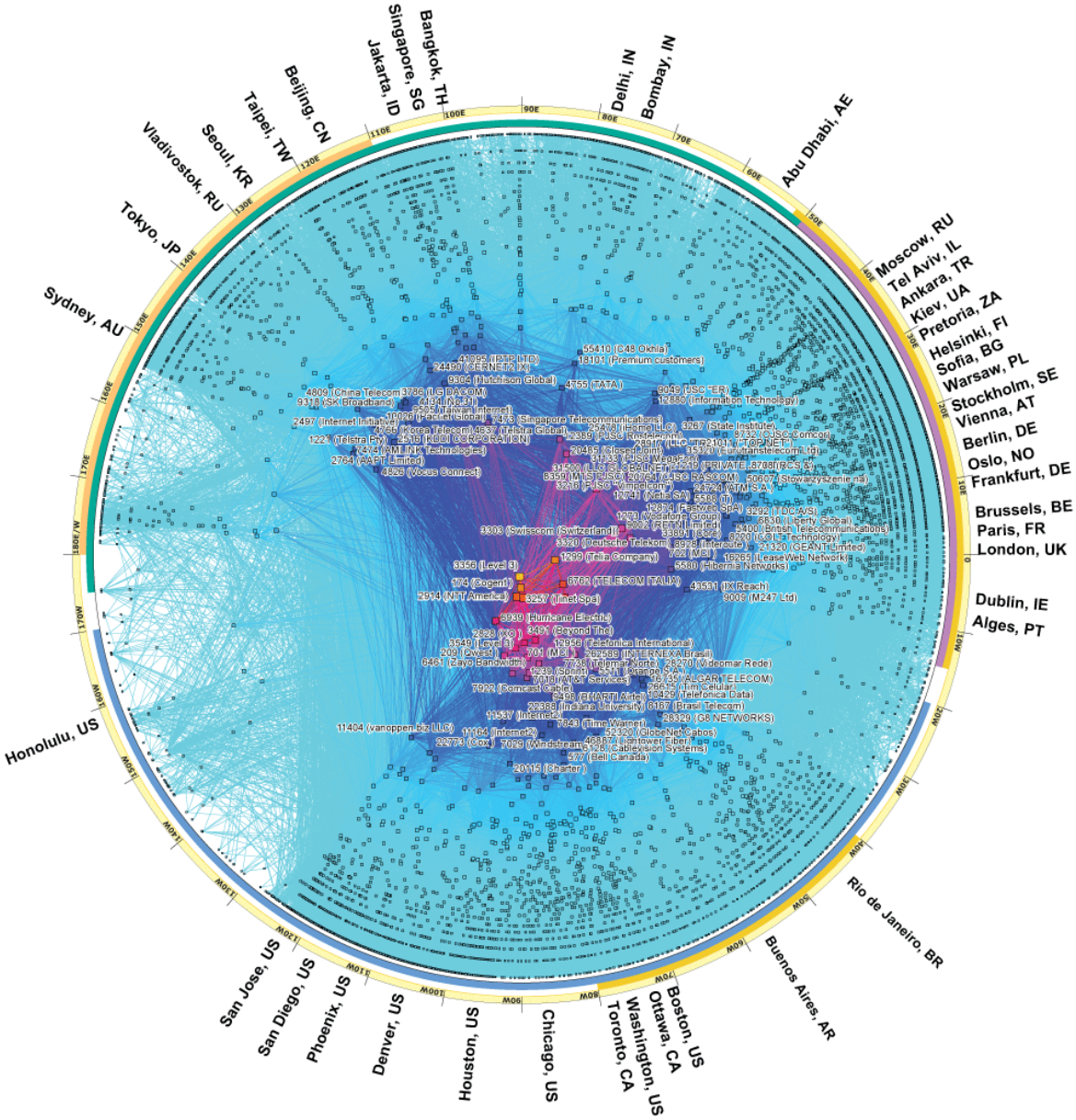


Figure 3: Visualization of the core IPv4 Internet topology in 2017 [30].

used in network attacks. Consequently, our perception of the Internet infrastructure is incomplete, and some network behaviors are still not well understood. For instance, the convergence of the *Border Gateway Protocol (BGP)* is still at the center of a lot of researches [62, 73, 151]. *Internet Control Message Protocol (ICMP)* filtering is important for optimizing congestion and bandwidth inference tools [82, 105], but its modeling is a complex problem which is still not optimally solved. Finally, the interpretation and correction of some inferred Internet models are still open questions [52].

In order to fill this lack of knowledge, the research community relies on measurements performed over the worldwide network. The data can be collected in two different ways:

(i) *Passive measurement*: It consists in listening Internet traffic, without modifying it. This traffic can be captured by packet, with a packet analyzer such as `tcpdump`, or by flows, with a tool such as NETFLOW [34]. The second solution requires a more complex analysis of the packet's header (ports, source and destination IP addresses, etc.) in order to determine to which flow it belongs. In practice, the different measurement devices, as well as the collected information, can be managed thanks to specific network protocols, such as the *Simple Network Management Protocol (SNMP)* [26]. A possible application of passive measurements is the inference of a traffic matrix. It reflects, over a given period of time, the volume of traffic observed between each pair of entry and exit border routers of a network. This matrix can be used in multiple ways. For example, it allows to plan the capacity of a core network [33], to detect anomalies [138], to optimize the routes inside the domain [131], to measure the amount of traffic sent by a client for billing [104, Chapter 15], or even to design new protocols [127].

(ii) *Active measurement*: This kind of measurement relies on probe packets injected into the network. Most of the time, these probes trigger a reply message from the target, or from intermediate routers on the path towards the destination. The packets received by the source can then be analyzed in order to infer some characteristics of the network, such as its internal architecture [52], end-to-end delay [22, 65], routing loops [78, 118, 150], path stability [4, 141], etc. In practice, well-known active measurement tools are `ping` [111] and `traceroute` [58].

One of the most popular research topics based on active measurements is the inference of the Internet topology. It consists in generating a graph representing how the different network elements are interconnected. Most of the time, this information is obtained from `traceroute` datasets. According to the desired granularity, the graph may be constructed at different levels [52]:

(i) *IP level*: a link corresponds to two successive IP addresses in a `traceroute` output.

(ii) *Router level*: the nodes represent network devices (routers, end-hosts, ...). An alias resolution technique [88] is then necessary in order to identify the IP addresses belonging to the same device in the `traceroute` dataset.

(iii) *AS level*: each node is associated to an AS, such as in Figure 3. This level allows to study the global connectivity of the different organizations in the Internet.

A lot of studies are based on graphs representing the Internet topology. Although heavily questioned [39, 93, 107, 146], one of the most popular papers in the field is the one of Faloutsos et al. [61] highlighting the power-law shape[2] of the node degree distribution (i.e., the number of neighbors of a network device). However, inferring a topology from `traceroute` measurements is a complex problem. In practice, network equipments, protocols, and specific configurations may bias `traceroute` outputs. For instance, load balancers may transmit successive `traceroute` probes via different paths. In this case, consecutive hops in the output are not necessarily neighbors in reality. Then, anonymous routers do not send back any ICMP message back to the source when they drop a packet. The measurement tool is thus unable to identify them, and a gap (a $*$) is observed in the output. In addition, Layer-2 devices are invisible to `traceroute`, while they may interconnect multiple routers in practice. Finally, the reverse path (i.e., from destination to source) is not necessarily identical to the forward one (i.e., from source to destination) due to BGP path asymmetry.

The graphs inferred from biased data may contain errors that may influence the conclusions of some studies. The research community has already suggested several `traceroute` improvements that take into account several network behaviors. For example, `paris-traceroute` [10, 42] ensures that all probes follow the same IP route between the source and the destination, even in case of load balancers. In addition, its *Multipath Detection Algorithm (MDA)* extension [11] is able to enumerate all load-balanced paths on this route. On its side, `tracebox` [50] allows to identify the middleboxes inside a network. These devices perform operations "apart from normal, standard functions of an IP router" [25], and may thus have an effect on `traceroute` outputs. Another tool, `tracenet` [59], can reveal all the interfaces connected to the different subnetworks crossed on the route towards the destination. Mérindol et al. [107] have also suggested a method for identifying Layer-2 devices interconnecting routers[3].

Other sources of error in graphs representing network topologies are hidden routers. Indeed, these devices are completely invisible in `traceroute` outputs. Two successive hops in a trace may thus appear as direct neighbors, while they are not in practice. Consequently, the inferred graph may contain false links, as well as nodes with an artificially high degree, and the resulting models and study results may thus be biased. Only a few researches suggested methods to detect this kind of routers. On the one hand, Marchetta et al. use the ICMP Timestamp option

[2] "A power law is a relationship in which a relative change in one quantity gives rise to a proportional relative change in the other quantity, independent of the initial size of those quantities" [17].

[3] This IGMP-based technique is now deprecated.

in their tool DRAGO [103] to infer the presence of hidden devices. On the other hand, Sherwood et al. exploit the IP Record Route option in their Internet cartographers PASSENGER [136] and DISCARTE [135] in order to reveal them. Marchetta et al. later extended this technique with ICMP Parameter Problem messages [102].

In their papers, Sherwood et al. suggested that hidden routers were probably caused by *MultiProtocol Label Switching (MPLS)* [130]. However, this question was never investigated further. MPLS will thus be at the core of this thesis: its infrastructure will be studied through active measurements, and new techniques will be introduced in order to reveal routers that are invisible in `traceroute` outputs due to particular MPLS mechanisms.

Motivations

Originally, MPLS was designed in order to increase the speed at which packets are transferred in a network. Briefly, MPLS routers determine the next hop on the path to the destination based on a short fixed-length label inserted in the packet's header. The set of MPLS routers forming a route in a network is called MPLS tunnel.

MPLS is deployed on the Internet today for many other purposes. For instance, an *Internet Service Provider (ISP)* may use it in order to interconnect different sites of a client, creating so a *Virtual Private Network (VPN)* [110]. MPLS also allows operators to optimize the performance of their network thanks to *Traffic Engineering (TE)* solutions [15, 139, 153]. The protocol is also often used in transit networks to carry the traffic between border routers. In this way, the internal architecture can be simplified, as only border routers can be aware of external destinations. A last example of use is *fast reroute* [115], where backup MPLS tunnels are pre-configured to take over in case of failure of the main tunnel.

Even if MPLS is largely used by network operators [53], it is not well investigated by the research community. Prior to this thesis, only two studies were specifically conducted on the identification of MPLS tunnels in `traceroute` data. First, Sommers et al. [137] evaluated the deployment of the protocol on the Internet based on three years of measurements. However, they only focused on tunnels that are visible in `traceroute` outputs. Hidden MPLS devices were thus not taken into account. Second, Donnet et al. [53] presented a measurement-based classification of MPLS tunnels. They considered all MPLS behaviors that may be encountered in the Internet. They also gave a first explanation on the origin of MPLS routers that do not appear in `traceroute` outputs, and tried to statistically quantify them. However, the authors were unable to reveal them. Moreover, some of the suggested models contain errors, and lack of precision.

Consequently, the current state-of-the-art view of MPLS on the Internet is incomplete, in terms of deployment and use by network operators, but also in terms of identification in `traceroute` measurements. Indeed, the study of Sommers et al. only focused on visible MPLS

tunnels, while Donnet et al. only gave an approximative estimation of the presence of hidden MPLS routers in the worldwide network. In addition, the revelation techniques suggested by Sherwood et al. and Marchetta et al. suffer from two major drawbacks. First, not all routers in the Internet support the IP Record Route and ICMP Timestamp options. Then, the Record Route option is limited to the first 9 hops of a trace, and restricts thus the research of hidden devices at first part of the path to the destination.

Main Contributions

This thesis aims at establishing a first detailed characterization of MPLS by answering the two following questions:

(i) *How is MPLS used and deployed by network operators?* The answer to this question will be brought through an *architectural* analysis of MPLS tunnels observed on today's Internet, and will allow to improve the culture and knowledge of the research community on the protocol.

(ii) *How can we reveal the network devices hidden by MPLS to* `traceroute` *explorations?* This question will be answered based on new *measurement* techniques that allow to infer the presence of hidden MPLS tunnels in `traceroute` data, and expose their content.

More specifically, the answers brought by this thesis to the first question (*architecture*) are multiple:

- As it will be shown in the different chapters of this thesis, MPLS behaviors may vary from one router vendor to another. In this way, identifying the manufacturer of the Internet devices appearing in `traceroute` measurements can help to recognize specific types of MPLS tunnels, and even to guide new techniques for revealing hidden MPLS routers. One of the contributions of this thesis is a *lightweight fingerprinting technique* [147] that allows to determine the manufacturer and operating system of routers based on `traceroute` and `ping` data. This technique will allow to confirm that CISCO devices are the most deployed in the Internet. It will also be applied on MPLS networks to perform their structural analysis, and show that JUNIPER devices seems more used in MPLS domains that in native IP ones. Fingerprinting will also be used when revealing hidden MPLS routers.

- MPLS is deployed in the Internet for multiple purposes. As already mentioned in the previous section, the protocol can be used to transport transit traffic between border routers of a domain. However, while a packet may be transferred along the shortest IP path, a network operator may decide to perform traffic engineering in his network. In this way, two data flows may be considered differently by the routers, and even follow separate routes between the two same border devices. In this thesis, we will study the use of MPLS in

transit networks. More specifically, the *Label Pattern Recognition (LPR)* algorithm [144] will be introduced. It allows to determine how MPLS is used in a given AS. In short, it is able to discriminate between a standard MPLS behavior (i.e., all packets follow the same route between two border routers), traffic engineering, and *Equal-Cost Multi-Path (ECMP)*. Based on a `traceroute` measurement campaign, we will show that the use of MPLS varies greatly from an AS to another. Even if the standard configuration of MPLS is the most widespread, we will see that several operators make use of traffic engineering in their network.

- The growth of the Internet continues day after day. More and more devices are interconnected over the worldwide network. A few years ago, the free pool of IPv4 address space of the *American Registry for Internet Numbers (ARIN)* [8] depleted [5]. Even if IPv4 is still the most widespread protocol, the transition to IPv6 is now a reality. The question of the *deployment and use of MPLS in IPv6* will thus be discussed in this thesis [145]. More specifically, we will see that MPLS is used differently in IPv6 networks than in IPv4 ones. Indeed, based on a dataset of `traceroute` measurements, we will show that it is mostly deployed in dual-stacked core networks where IPv6 versions of some important protocols are still not available, or where MPLS tunnels that already carry native IPv4 traffic can be exploited to transport IPv6 packets too.

The answers brought by the thesis to the second question (*measurements*) are twofold:

- Hidden MPLS routers may be the cause of multiple issues in graphs representing network topologies. Their identification is thus important to avoid biases in Internet models. A large part of this thesis aims at presenting different *new techniques to infer their presence, and reveal them when possible* [146]. These techniques were validated in a virtual environment simulating networks with real router OSs, and cross-validated based on on a large-scale measurement campaign targeting suspicious networks. Thanks to a `traceroute` dataset, we will show that invisible MPLS tunnels are not rare in the worldwide network. Moreover, we will demonstrate that the inference of basic Internet graph properties, such as path lengths and node degrees, can be improved when hidden devices are taken into account.

- Donnet et al. [53] were the first to present a measurement-based classification of MPLS tunnels. However, some of their models are not perfect and lack of precision. The last contribution of this thesis is the correction of these models, and their integration in *Trace the Naughty Tunnels (TNT)*, a tool able to identify all kinds of MPLS tunnels along the path between a source and a destination [143]. It integrates the techniques for inferring the presence of invisible tunnels, and revealing their content when possible. Based on the results of a large-scale measurement campaign performed with `TNT`, we will also update the state-of-the-art quantification of MPLS tunnels in the worldwide network.

Outline

This document is divided into three parts. The first one gives the required technical background for this thesis. More specifically, Chapter 1 describes the origin and functioning of MPLS in detail, as well as its most common fields of application on the Internet. On its side, Chapter 2 presents the state of the art on MPLS researches, focusing on the classification of Donnet et al. [53] used as a reference in this thesis.

The second part focuses on the architectural aspect of MPLS. More specifically, Chapter 3 introduces the lightweight fingerprinting technique [147] allowing to identify the manufacturer and operating system of routers based on `traceroute` and `ping` measurements. Then, Chapter 4 evaluates the use of MPLS in transit network thanks to LPR [144]. Finally, the study of MPLS in IPv6 networks [145] is conducted in Chapter 5.

The last part of the document describes several techniques allowing to identify MPLS tunnels in `traceroute` measurements. In particular, Chapter 6 presents inference techniques able to signal the presence of PHP invisible MPLS tunnels[4] in a trace, as well as two methods to reveal their content, when possible [146]. Eventually, Chapter 7 introduces and evaluates `TNT` [143], the tool able to recognize any kind of MPLS tunnel on the path towards a specific destination.

Finally, this documents ends with a summary of the different contributions, as well as a discussion on improvements and future works.

[4]The concept of *Penultimate Hop Popping (PHP)* will be introduced in Section 1.5.5.

Part I

Background

CHAPTER 1

MULTIPROTOCOL LABEL SWITCHING

This chapter is a description of *MultiProtocol Label Switching (MPLS)* [130]. It introduces the concepts of MPLS forwarding (data plane), as well as the architecture of MPLS networks (control plane), and shows their advantages compared to conventional IP networks. Different use cases of the protocol are also presented, to demonstrate its interest on today's Internet.

1.1 Introduction

Each day, millions of devices exchange data on the Internet [125]. All these devices, also called *end hosts*, are interconnected thanks to *routers*. Routers allow the transmission of the data between the source and the destination hosts. During a communication between two devices, packets containing the information to exchange follow a specific path composed of several routers, as shown in Figure 1.1.

A router is organized following three conceptual operational planes [57]. The first plane, known as the *forwarding plane*, or *data plane*, is responsible of transmitting packets from an input interface to an output interface. The *control plane* is in charge of determining how the data should be forwarded. Finally, the *management plane* manages the device through its connection to the network.

The matter covered by this chapter is related to the control and data planes. These two planes ensure that a packet received by the router will be transferred correctly to the next hop (i.e., the next router) on the path to the destination. The set of operations performed by the device to transmit a packet is called *packet forwarding*, and is based on the packet header (typically, at the network layer).

MPLS is a packet forwarding technology originally designed to increase the speed at which the data is transferred along a path. The term *MultiProtocol* is used to specify that the techniques are applicable to any protocol layer. In this thesis, we will however focus on the network layer.

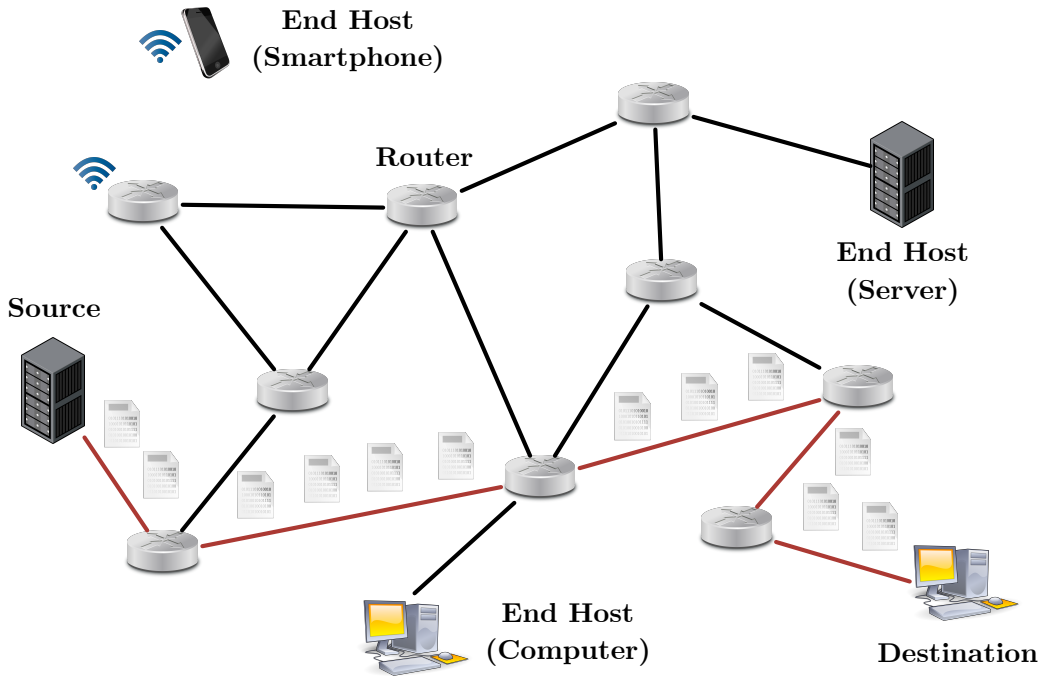


Figure 1.1: Data exchange between a source and a destination on a computer network. The path followed by the data is highlighted in red.

1.2 IP Forwarding

Today's Internet can be seen as a network of organizations. Each organization consists in a collection of routers and local networks. These routers are aggregated into regions, called *Autonomous Systems (ASs)*. The different ASs are interconnected, and have their own independent administration, as shown in Figure 1.2. Each time a packet arrives at an input interface of a router, the device must take a forwarding decision. This decision is based on the information contained in a table called *Forwarding Information Base (FIB)*, or more simply, *forwarding table*. In order to have a global connectivity, each router must have an entry in its table that matches the destination of each packet.

Internet Protocol (IP) [121] is the most popular network layer protocol deployed on the Internet today. It allows devices to communicate with each other. Each host connected to a network has its own and unique IP address. It is used to locate the device in the network. *IP forwarding* is based on these addresses. All the routers are able to forward packets thanks to their destination address.

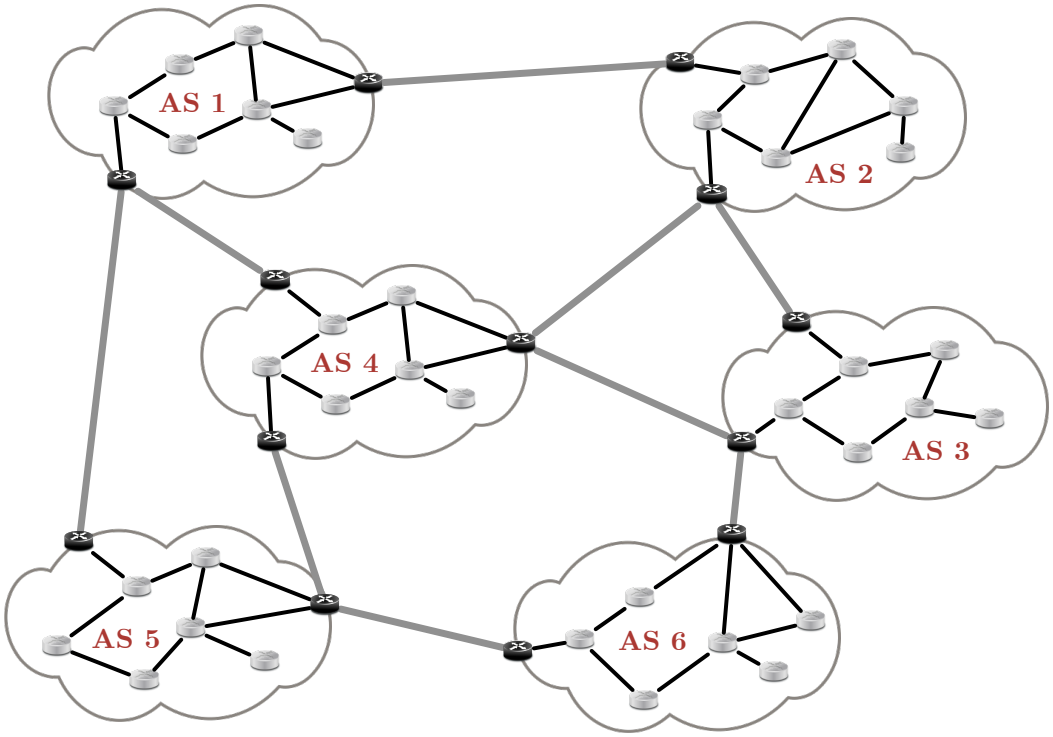


Figure 1.2: Network of interconnected ASs. Routers at the edges of a network (black routers) are in charge of connecting the AS to its neighbors.

As billions of devices are interconnected in practice, creating an entry for each destination in the IP forwarding tables does not scale.

In practice, the *Internet Assigned Numbers Authority (IANA)* [81] is in charge of allocating IP addresses to the different organizations. Each AS receives one or several *prefixes* defining a set of IP addresses that can be administrated. As a prefix is associated to a unique AS, it can be used to populate the forwarding tables. In this way, a single entry in a table is sufficient to forward all the packets having the same destination prefix. Note that multiple prefixes can be aggregated in order to form larger prefixes, and reduce the size of the tables.

Each router in the network builds its own forwarding table in two steps. This set of operations occur in the control plane. The device runs a routing algorithm (intra-domain, such as *Routing Information Protocol (RIP)* [101], *Open Shortest Path First (OSPF)* [109], and *Intermediate System to Intermediate System (IS-IS)* [114], or inter-domain, like *Border Gateway Protocol (BGP)* [126]) in order to generate a *Routing Information Base (RIB)*, also known as *routing table*. This table contains all the possible routes to each known prefix. Then, the best route to each prefix is selected and injected in the FIB. Storing multiple paths to a single destination in the RIB allows to choose an alternative route in case of failure on the selected route.

Once the FIB is complete, the router is able to forward the data. Each time it receives a

packet, it finds the longest prefix in the forwarding table that matches the packet's destination IP address. The corresponding entry specifies the output interface and the next hop to which the packet must be sent, as illustrated in Figure 1.3.

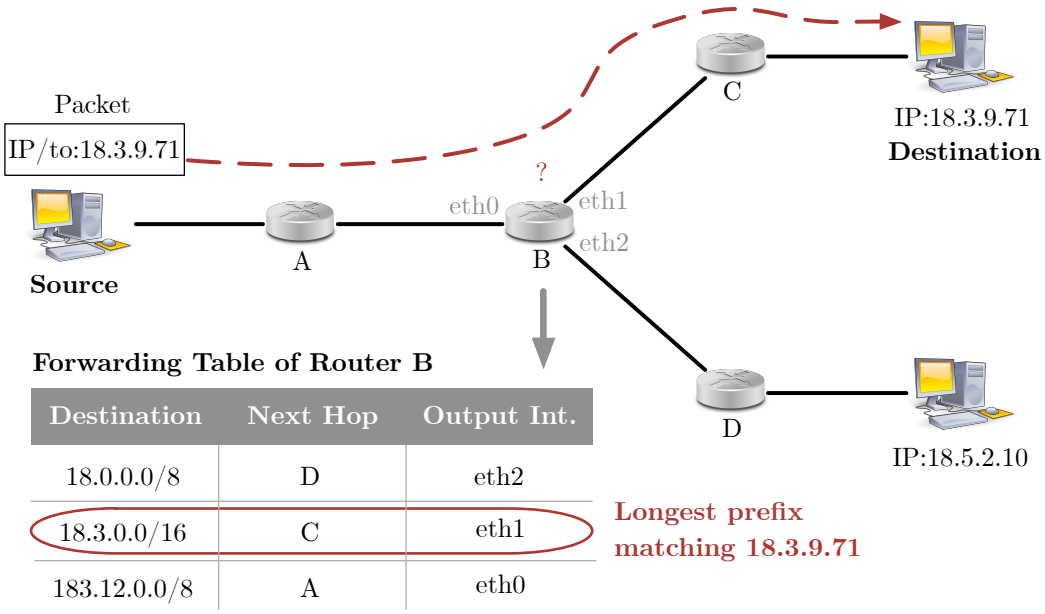


Figure 1.3: IP forwarding. Router B identifies 18.3.0.0/16 as the longest prefix matching the packet's destination 18.3.9.71. The forwarding table specifies the packet must be forwarded to C via the output interface eth1.

The longest prefix matching is a difficult operation performed based on a specific data structure called *trie* [133]. A trie, also called *prefix tree*, is a search tree allowing to store strings over a specific alphabet. This kind of data structure is different from a binary search tree, as the nodes do not store keys. Each branch of a trie represents a possible character of the key. The position of a node in the tree determines thus the key it is associated with, and all its children share this key as a common prefix. As shown in the example in Figure 1.4, in the case of forwarding tables, the trie stores the different routing prefixes. A prefix defines a path from the root to a node. Each node, leaf or internal, determines the next hop the packet must be forwarded to, if a prefix ends on it.

Although IP is heavily used in computer networks, the forwarding based on this protocol suffers from a few drawbacks. Indeed, even if insertion and search operations in a trie are efficient, performing a longest prefix matching is more expensive that an exact string matching. Moreover, the memory space requirements to store the trie structure is large. Finally, forwarding decisions at each router are independent, as the next hop is determined based on the packet's destination. The source does not have any impact on the route followed by the data. This kind of network, called *packet-switched network*, makes then difficult the deployment of solutions

Destination: 201.22.110.12
201_b: **11001001**

Prefix	Next Hop
01*	B
010*	G
0110011*	C
0111*	D
10*	H
11000*	F
11001000*	I
11001001*	E
1101*	A

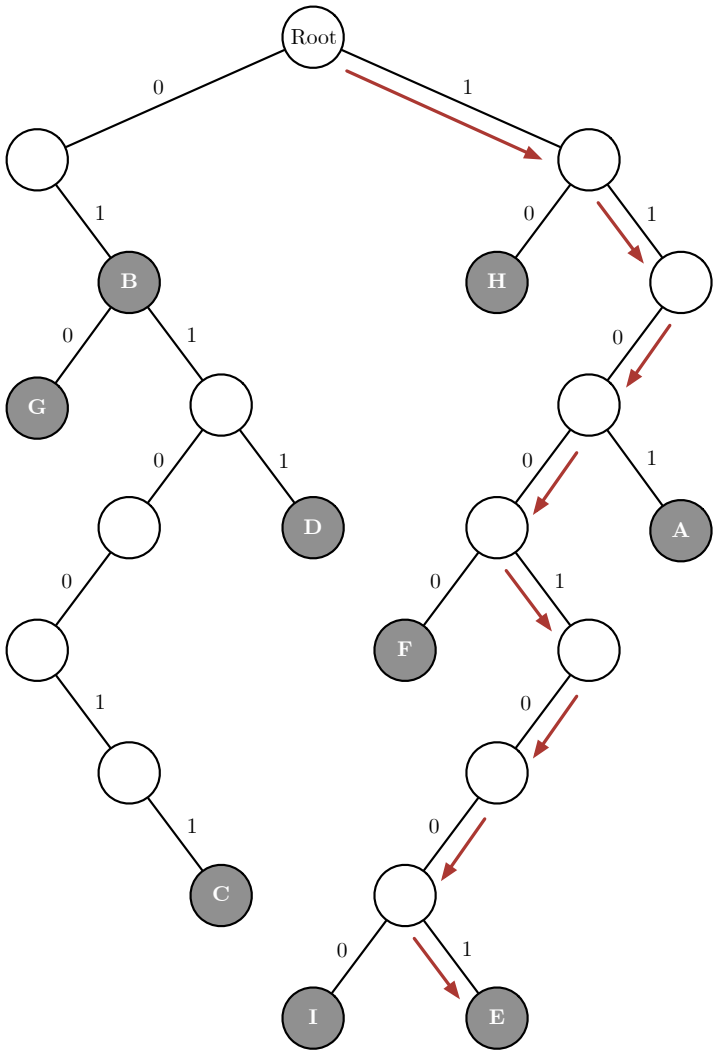


Figure 1.4: Longest prefix matching with a trie. The red arrows shows how the tree is traversed based on the binary notation of the destination 201.22.11.12. The search ends when leaf E is reached, meaning that the longest prefix is 201.0.0.0/8 (i.e. 11001001*), and that the next hop is E.

allowing to use different paths based on other type of information than the destination of the packets.

1.3 Forwarding Equivalence Class

The notion of *Forwarding Equivalence Class (FEC)* introduces more flexibility in the forwarding process. It was created in order to allow routers to transmit packets based on other information than their destination IP address. A FEC can be defined as the set of packets forwarded in the same way. For example, a class can gather packets with the same source and destination, or packets that must be considered in priority compared to others. The FIB of a router is then only populated with one entry for each FEC. With this new concept, packet forwarding at a router can be seen as associating the packet to a FEC, and then, based on the class, determining the next hop on the path. This is illustrated in Figure 1.5. If the principle of FEC is applied to IP forwarding, each class would gather the packets having the same destination IP prefix.

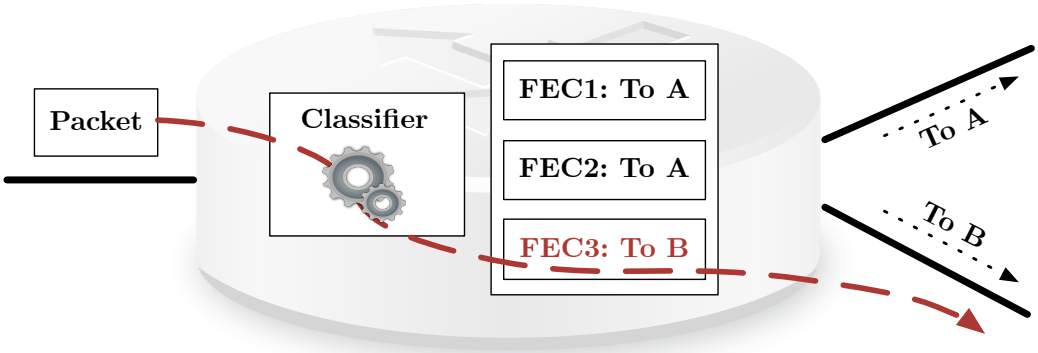


Figure 1.5: Packet forwarding based on FECs. The router associates the packet to class FEC3, which specifies it must be forwarded to B.

1.4 Virtual Circuits

In a packet-switched network, the resources are not reserved, but used on demand. As a consequence, packets may have to be stored in a wait buffer before accessing the communication link. This technology is not the only one allowing to transport data on a network. Other solutions exist, such as *circuit switching* [92], used in traditional telephone networks.

In this architecture, the resources, such as the available bandwidth, are equally divided into pieces. When two end hosts want to exchange information, an end-to-end connection, called *circuit*, is established from the source to the destination. The different forwarding devices on the path maintain a state, and reserve one, or several chunks of resource for the communication. The

sender is then ensured to transmit its data at the guaranteed constant rate. When the transfer is completed, the network forgets the circuit, and reallocate the corresponding resources. A simple example of circuit-switched network is shown in Figure 1.6.

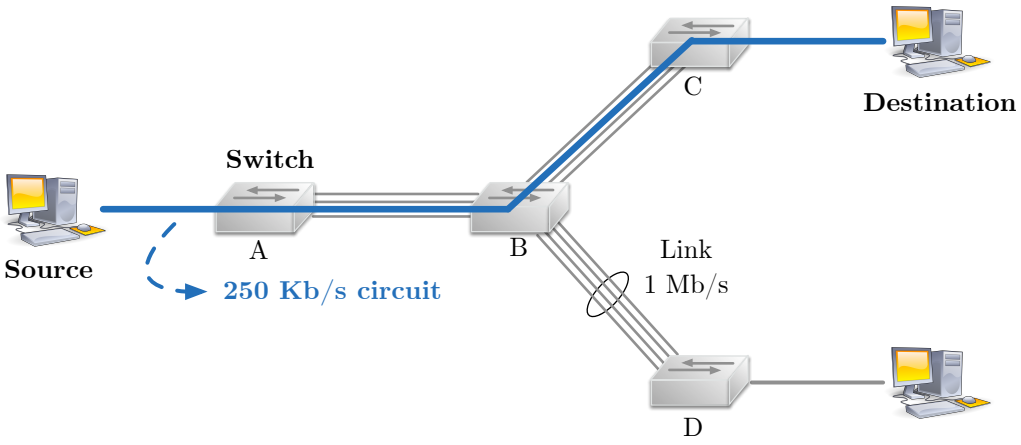


Figure 1.6: Circuit-switched network. Four switches are interconnected by three links having a capacity of 1 Mb/s. Each switch is able to support 4 simultaneous circuits, meaning each connection can receive 250 Kb/s of rate to transmit data. An end-to-end connection (blue line) has been established between the source and the destination, leaving only three available circuits on the links between switches A and C.

Since the devices store information for each connection, the destination address is not needed to forward the data. However, the establishment of a communication introduces delay, as all switches on the path must be notified of the new exchange, and must then allocate the necessary resources before the data could be sent. Moreover, once bandwidth has been reserved, it can not be shared with another circuit, even if it is not used completely. It may lead to an inefficient sharing of the resources, especially in the presence of bursty flows.

Virtual Circuits (VCs) [92] allow to emulate a kind of circuit-switched architecture over a packet-switched network. The technology tries to keep the advantages of both techniques. For each data exchange, a virtual connection is established between the two end hosts, simulating a dedicated physical link between them. The routers on the path keep track of the different circuits crossing them. In a VC, a short number is assigned to each link along the route. This identifier is inserted in the packet header, and compared by each router to the values available in its forwarding table. The entry matching the number determines how the packet must be forwarded. The destination is not taken into account anymore. The VC identifier in the packet header is updated at each hop, according to the content of the forwarding tables, as illustrated in Figure 1.7.

At each new transmission, a setup phase determines a path to the receiver, and the different values for the identifier. An entry is then added in the forwarding table of the routers in the

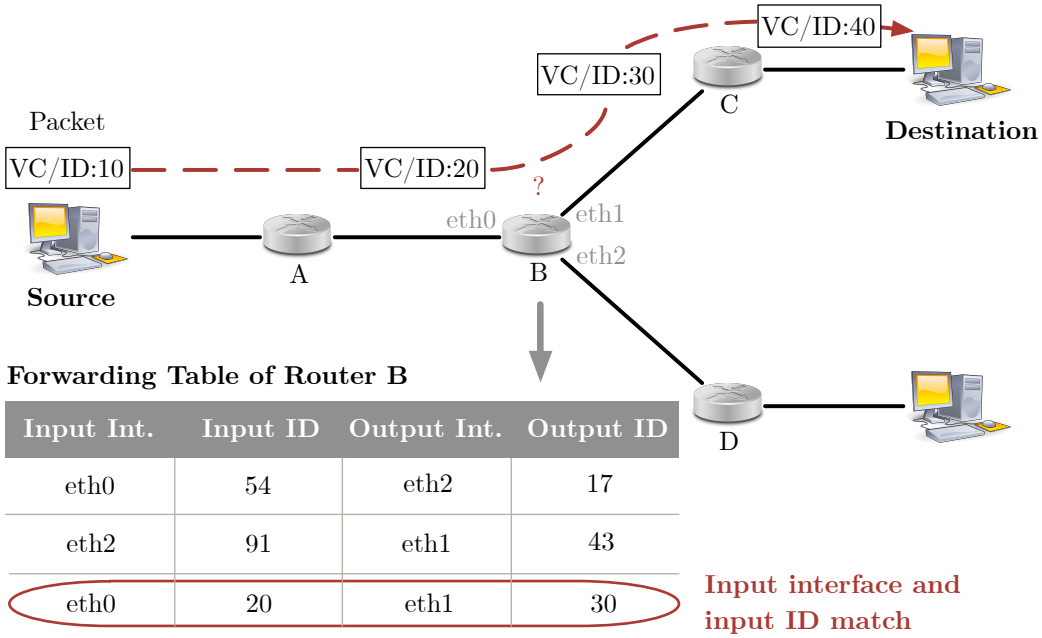


Figure 1.7: Packet forwarding with virtual circuits. Router B receives a packet on its input interface eth0 with the VC identifier set to 20. According to its forwarding table, the packet must be forwarded via the output interface eth1, after having replaced the VC number in its header by 30.

circuit. At the end of the transfer, the devices are notified that the VC no longer exists, and the corresponding entries in the forwarding tables are deleted. The resources are shared between the virtual connections, without waste. However, a reservation mechanism may be applied to guarantee a certain level of quality of service to some applications.

1.5 MPLS Architecture

In an IP network, routers must perform a longest prefix matching each time a packet must be forwarded. One of the main advantages of using virtual circuits is that identifying the next hop on the path requires only to compare short fixed-length numbers. This operation needs much less computing resources than a longest prefix matching. As a result, the forwarding speed in a virtual circuit is improved compared to conventional IP networks.

At then end of the nineties, industries started thinking of a way to bring the concept of virtual circuits to the conventional IP network [92]. The idea was not to replace the existing IP architecture, but to improve it by upgrading routers with the capabilities of selectively labelling packets, and forwarding them based on fixed-length values, when possible. The MPLS protocol [130] is the technological answer allowing this interoperability.

1.5.1 MPLS Network

Referring to the packet forwarding definition, each time an IP datagram enters an MPLS domain, it must be associated to a FEC. The protocol allows to create classes based on other fields than the destination IP address. The FEC is encoded as a short fixed-length value, called *label*, and is sent alongside the packet in the network. Similarly to the virtual circuit approach, this label is used by the internal routers in order to perform the forwarding operations, and is updated at each hop. A packet is associated to a FEC only once, at the entrance of the network. The first router in the domain may then define the forwarding behavior inside the entire network.

MPLS works hand-on-hand with IP, using IP addressing and routing. Both protocols are linked to the data and control planes of a router, as shown in Figure 1.8. Similarly to the virtual circuit architecture, a specific forwarding table working with labels is generated for each router. This table, named *Label Forwarding Information Base (LFIB)*, is the MPLS version of the IP FIB, and is generated based on the *Label Information Base (LIB)*. The LIB is populated by a *label distribution protocol*[1] using the data contained in the RIB. Similarly to the RIB that gathers all existing routes to each known IP prefix, the LIB is aware of all possible FEC to label bindings. Only the best bindings are selected to generate the LFIB. Unlike the virtual circuits approach, the router builds its tables before starting forwarding packets. The virtual paths are also permanent, meaning that they are not removed from the LFIBs when a connection is closed.

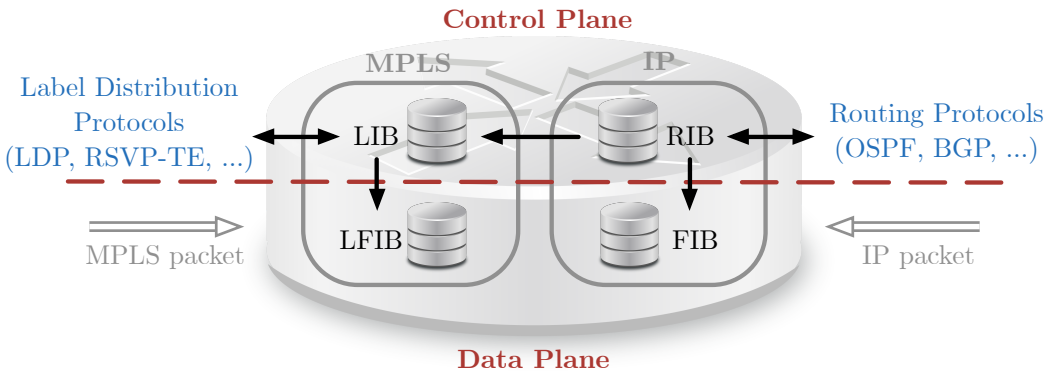


Figure 1.8: Data and control planes in an MPLS router.

An MPLS network is built following a well-defined structure illustrated in Figure 1.9. In the domain, the different routers are called *Label Switching Routers (LSRs)*. The routers at the edges of the network are also named *Label Edge Routers (LERs)*. When a connection must be ensured, a path is selected between two LERs. This path is also known as the *Label Switched Path (LSP)*. The entry and exit points, defined as the *ingress* and *egress LERs* respectively, and the LSRs between them, form an *MPLS tunnel* (shown in red in the figure, and composed of routers A to E).

[1]The label distribution protocols will be discussed in detail in Section 1.5.4

In a tunnel, the ingress router is in charge of inserting the first label in the IP packet, while the egress point performs the opposite operation. Finally, the first LSR that forwards packets based on labels only is named *First Hop (FH)*, and the last one is called *Penultimate Hop (PH)*. In an MPLS domain, multiple paths may exist between two LERs.

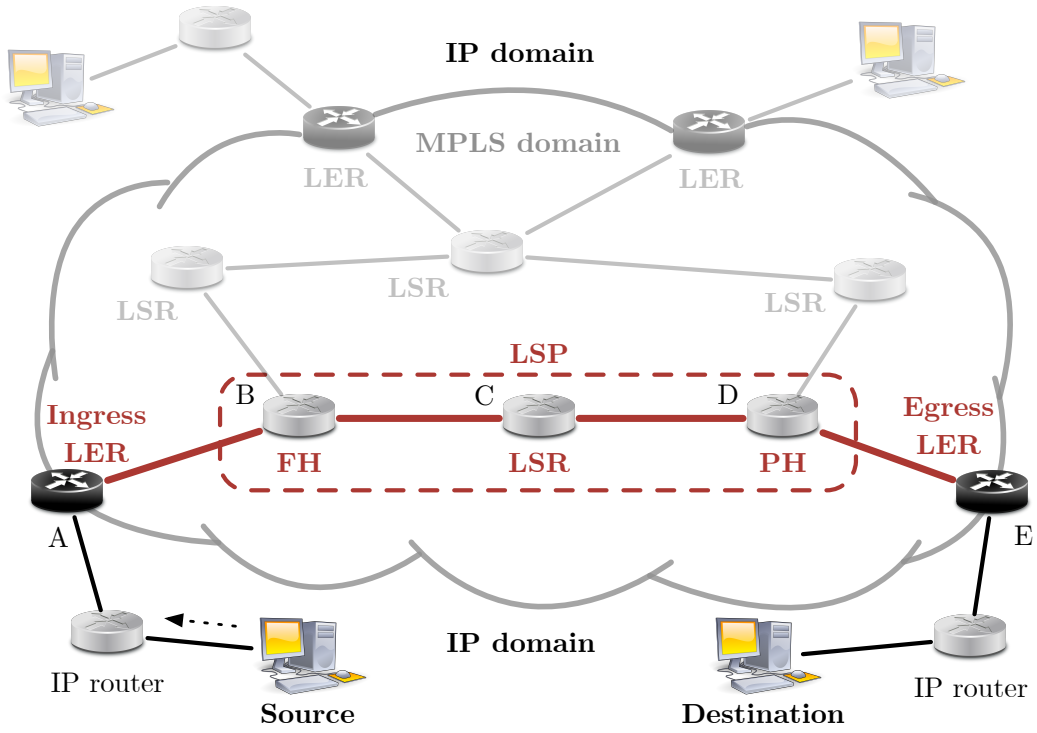


Figure 1.9: Structure of an MPLS network. An MPLS tunnel (routers A to E) forwards the packets sent from the source to the destination.

1.5.2 Label Stack Entry

In order to allow the forwarding operations, a label must be attached to each packet crossing an MPLS network. The value is stored in a small 32-bits header named *Label Stack Entry (LSE)* [129]. When MPLS is used at the network layer, as studied in this document, the LSE is inserted between the Ethernet and the IP headers, as shown in Figure 1.10.

An LSE is composed of four different fields. The first 20 bits are reserved for the label value itself. It may range from 0 to 1,048,575, noting that the first 16 values (0 to 15) are reserved, and have well-defined meanings. The next 3 bits encode the *Traffic Class (TC)* [6]. This field is used when a *Quality of Service (QoS)* [134] must be ensured for the connection, such as a minimum end-to-end delay, or a guaranteed minimum bandwidth, for example. A total of 8 different profiles are available. In some tunnels, multiple LSEs may be stacked in the same packet, as illustrated in Figure 1.11, allowing to create a more complex structure, such as a *Virtual Private Network*

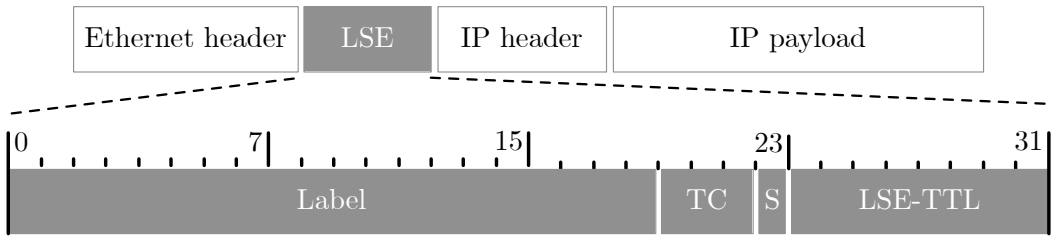


Figure 1.10: MPLS Label Stack Entry (LSE).

(VPN) [110], or a *6PE* architecture [47][2]. The *Bottom of Stack* field (S) is a single bit specifying if the current LSE is at the bottom of the stack. Finally, the last 8 bits are used to store a *Time To Live (TTL)* value. This field, also known as *MPLS TTL* or *LSE TTL* has the same aim as the well-known IP TTL. It ensures that the packet will not consume to much resources inside the MPLS domain. It is decremented by one at each hop. Once a packet with a TTL value of 1 is received by an LSR, it is dropped, and an *Internet Control Message Protocol (ICMP)* [120] `time exceeded` message is sent to the source to warn it the packet was destroyed before reaching its destination.

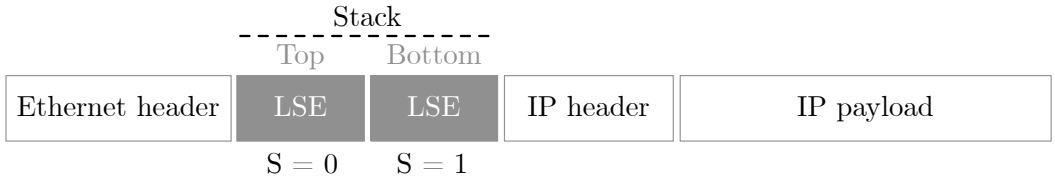


Figure 1.11: MPLS label stack.

1.5.3 Data Plane

In an MPLS network, each time an LSR receives a packet, it must take a forwarding decision based on the label values stored in the header, as illustrated in Figure 1.12. Similarly to IP, it consists in determining an output interface and the next hop on the path to the destination. This information is found in the LFIB at the entry matching the packet's top label value. The table also specifies an operation that must be applied on the label stack. Three possible actions, described in Figure 1.13, can be performed by an LSR.

The most common operation, mainly executed by the internal LSRs, is SWAP. It consists in replacing the label on top of the stack by a new one. The second action, PUSH, is performed by the ingress LER most of the time. The router simply replaces the top label on the stack (if any) with a new one, and pushes one, or several additional labels on top. As the ingress LER is in charge of

[2]These technologies will be discussed in Section 1.6.

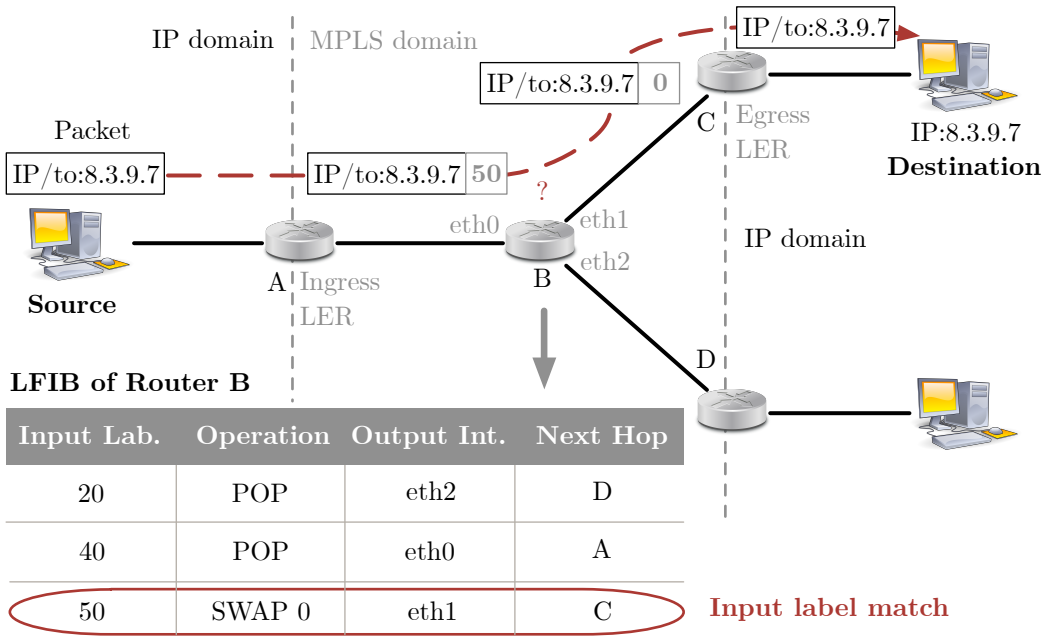


Figure 1.12: MPLS forwarding. Router B receives a packet on its input interface eth0 with the label value set to 50. According to its LFIB, the packet must be forwarded via the output interface eth1, after having replaced the label value by 0.

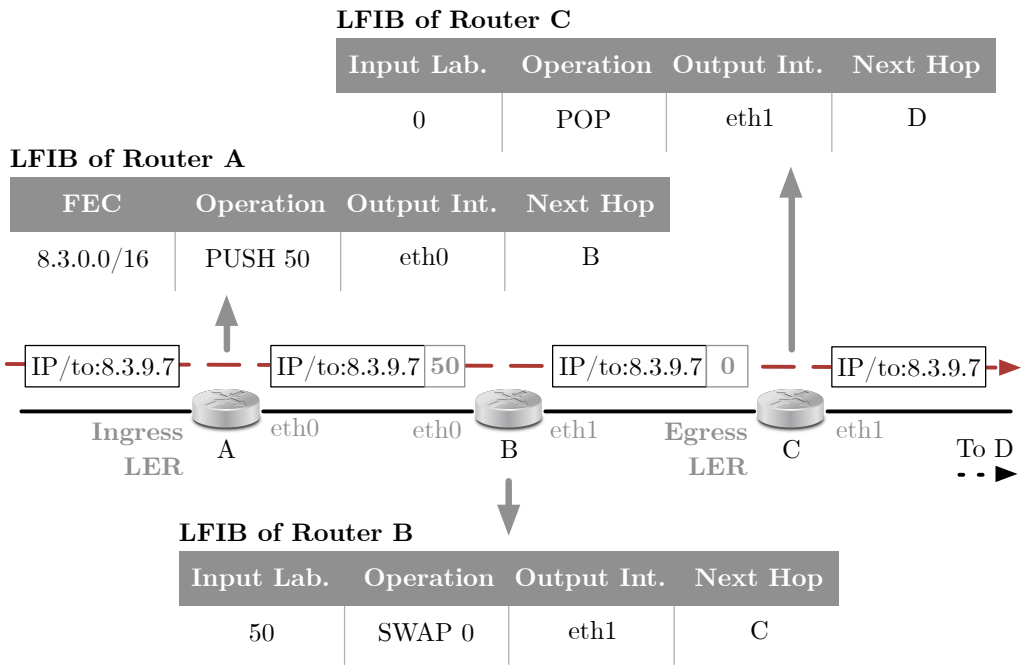


Figure 1.13: MPLS operations. Router A pushes label 50 in the IP packet, B swaps this label value by 0, and finally, C pops the LSE.

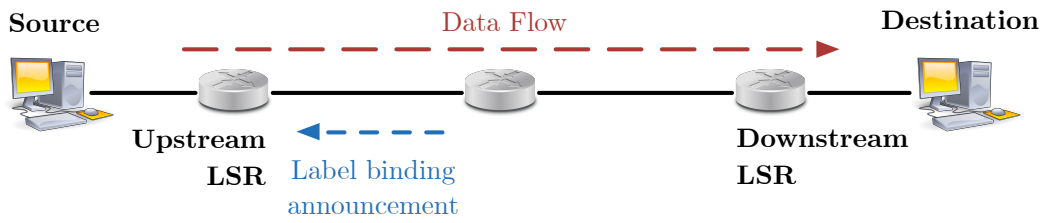
associating the packet to a class, and inserting the first label(s) in the header, its LFIB is slightly different, and includes the FEC. Finally, a router may also delete one, or all labels on top of the stack. This POP operation is mainly executed by the egress LER.

1.5.4 Control Plane

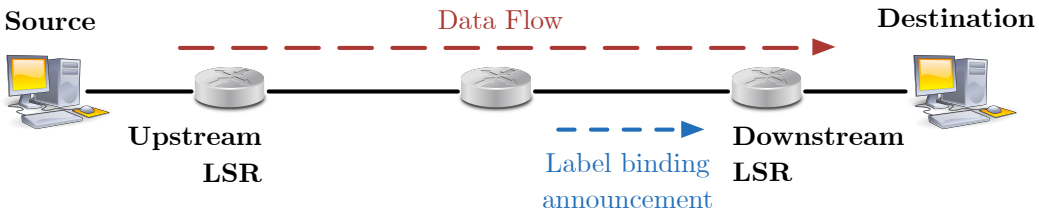
A router inside an MPLS domain stores its forwarding instructions in the LFIB, which is accessed according to the labels stored in the packet's header. This table is generated based on the LIB, containing all possible label bindings for each known FEC.

Each time a device learns a new class, it associates a label to it, and updates its LIB. These bindings are said *local*, and make sense for the router only, as the label values are chosen by the LSR itself.

In order to ensure that a packet will be directed to the proper destination, the router must use an outgoing label value that will be understood by the next LSR on the path. Each device must then announce its different local label bindings to its *upstream neighbor*, meaning the previous node in the MPLS tunnel. This assignment mode is called *downstream label allocation*. Another mode, named *upstream label allocation*, allows the router to send its mappings to the *downstream neighbor*. Both techniques are illustrated in Figure 1.14. A binding received from a neighbor is said *remote*, and is also inserted in the router's LIB.



(a) Downstream allocation.



(b) Upstream allocation.

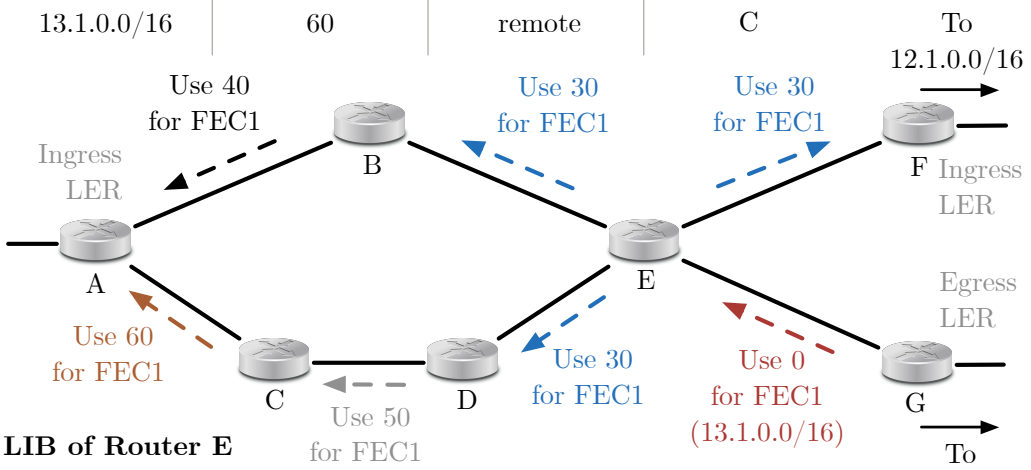
Figure 1.14: Label allocation modes.

In an MPLS network, FEC-to-label bindings may be distributed by the *Label Distribution Protocol (LDP)* [7]. This protocol uses the downstream label allocation, by default. As shown in

Figure 1.15, the egress LER associates first a label to a FEC and updates its LIB. This information is then sent to all the upstream neighbors, which in turn bind a label to the FEC, and update their LIB. These routers also inform all their own upstream neighbors of their association. These operations continue until reaching an ingress LER. As LDP works on top of an *Interior Gateway Protocol (IGP)* (OSPF, IS-IS, ...), all binding messages follow IP routes.

LIB of Router A

FEC	Label	Binding Type	Origin
13.1.0.0/16	40	remote	B
13.1.0.0/16	60	remote	C



LIB of Router E

FEC	Label	Binding Type	Origin
13.1.0.0/16	30	local	/
13.1.0.0/16	0	remote	G

Figure 1.15: Label Distribution Protocol (LDP). Egress LER G associates label 0 to FEC1, representing the destination prefix 13.1.0.0/16. The binding is then announced to neighbor E. Router E receives the remote binding, and decides to assign label 30 to the class. It updates its LIB, and signals its local binding to neighbors B, D, and F. The bindings continue until reaching an ingress LER (F or A).

A specific version of the *Resource ReSerVation Protocol (RSVP)* [23] can also be used to distribute labels amongst the different LSRs. In its common use, RSVP allows a node to request some quality of service for a data flow. It works in two steps, as shown in Figure 1.16. First, a PATH message is sent by the source to the receiver(s), along the same IP path(s) used by the data packets, determined by the underlying routing protocol. The different routers forwarding the message create a path state that records at least the previous router on the path to the destination. In the second step, each receiver sends back to the source a RESV message. This

message follows hop-by-hop the reverse data path. At each hop, the destination address of the message is updated with the IP address of the previous node, based on the path state previously created. The source address is also replaced with the address of the current node. In this step, the routers allocate resources in order to ensure the requested quality of service during the data transfer.

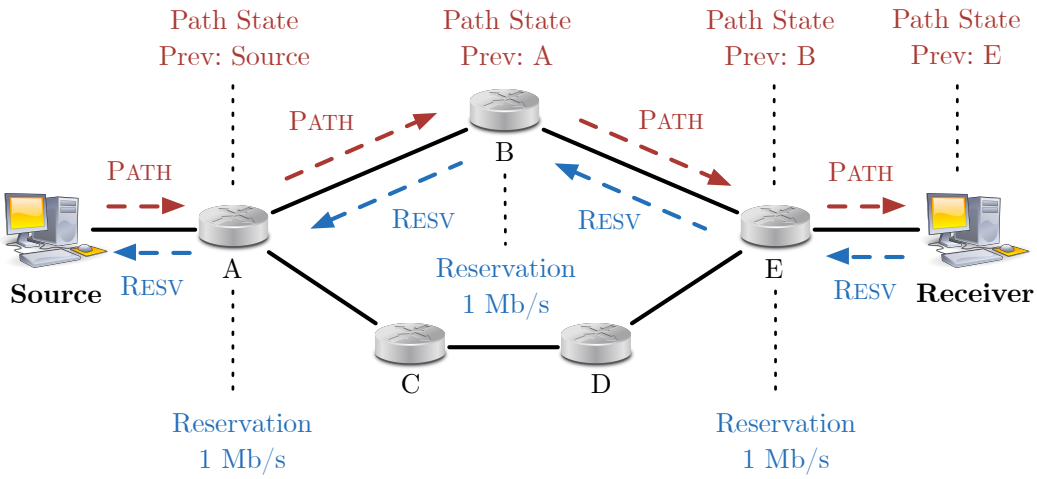


Figure 1.16: Resource ReSerVation Protocol (RSVP). The source sends first a PATH message to the receiver, following the IP path. Each node on the route creates a path state based on the message's content. In a second step, the receiver sends a RESV message back to the source to reserve a bandwidth of 1 Mb/s at each hop. This message follows the reverse path, using the state previously created at each router.

An updated version of RSVP, called *Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)* [12, 14], gathers a few extensions that allow its use in MPLS networks, as illustrated in Figure 1.17. In this version, the RESV messages can piggyback MPLS labels, meaning RSVP-TE is able to distribute the label bindings in the MPLS domain, following the downstream allocation mode. Moreover, with the *explicit route object* extension, the ingress LER, that sends the PATH message, can pre-calculate a LSP path that may be different from the standard IP route computed by the underlying routing protocol.

An operator running RSVP-TE can deploy a more complex MPLS architecture in his network, allowing different qualities of service depending on the FEC. Priority traffic may receive dedicated resources, or be deviated via a less congested path, for example.

1.5.5 Reserved Label Values

As described in Section 1.5.2, a wide range of labels can be selected by an LSR to map the different FECs. However, the first 16 values are reserved [129], and chosen in well-defined situations.

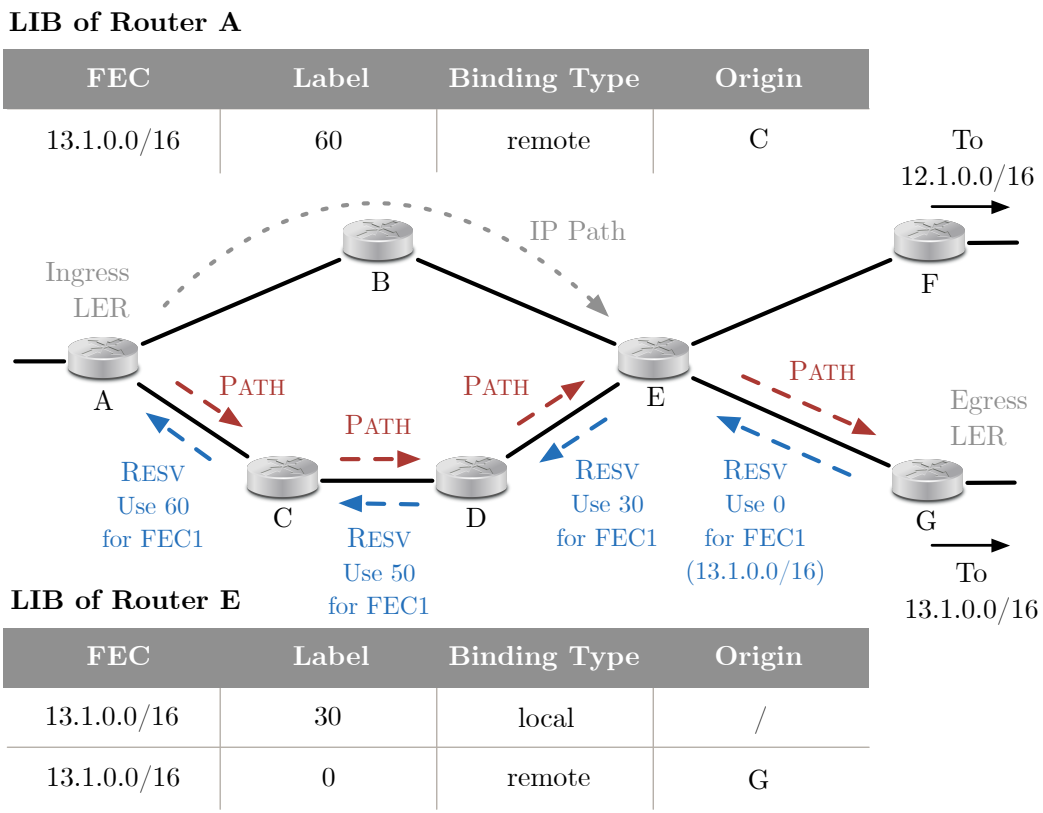
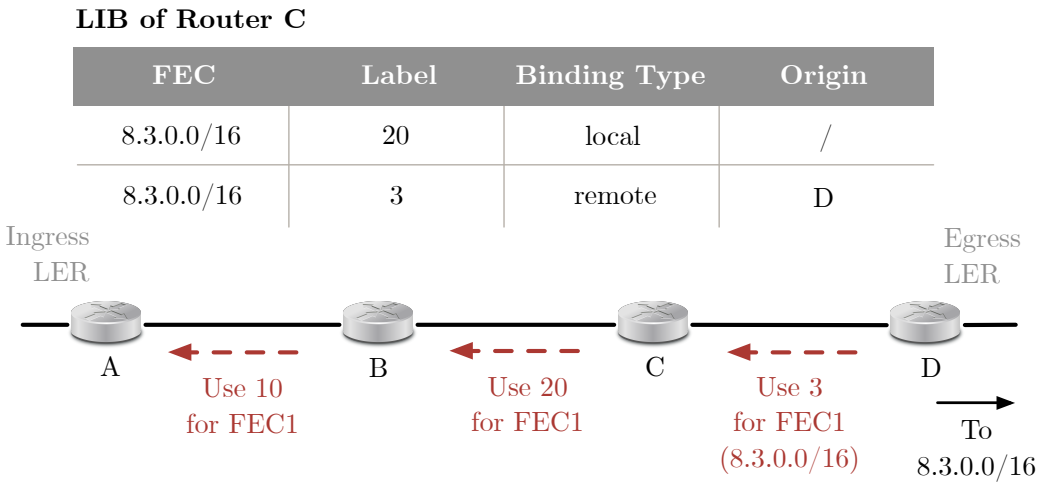


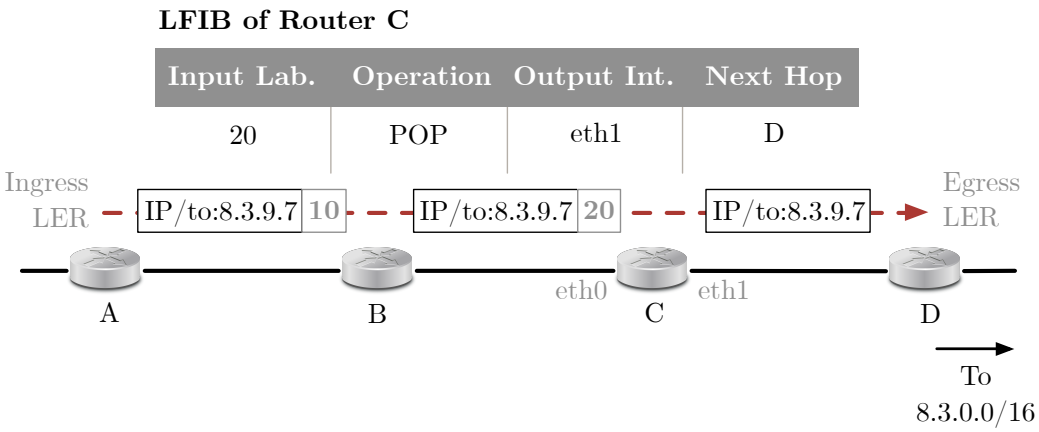
Figure 1.17: Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE). The ingress LER decides to send a PATH message to the egress LER without following the IP path specified by the underlying routing protocol. In the second step, the egress LER sends a RESV message back to the source. The different nodes use this message to announce their local label binding to their upstream node on the path.

A value of 3 identifies the *Implicit NULL Label*. It is said *implicit* because even if it may be assigned to a FEC, and announced to other routers, it is never written in a packet's header. When an LSR discovers that the next router associated 3 to a class, it pops the label stack before forwarding the packet. The top label is then never replaced by 3. The implicit label is announced by an egress LER in order to perform a *Penultimate Hop Popping (PHP)*, as illustrated in Figure 1.18. The traffic load received by this node is higher than any other LSR, as it may be connected to multiple ingress LERs, being then the exit point of multiple MPLS tunnels. If the label stack is popped by the previous node (the penultimate hop), the egress LER can directly forward the packet based on its destination IP address, and does not perform any MPLS operation. The computational resources needed to transmit the data are then reduced.

When a network operator decides to enable PHP, a simple IP datagram is transferred to the egress router. Even if the technique increases the forwarding efficiency, information that travels with the label stack is lost. In some situations, such as a quality of service based on the traffic



(a) Control Plane.

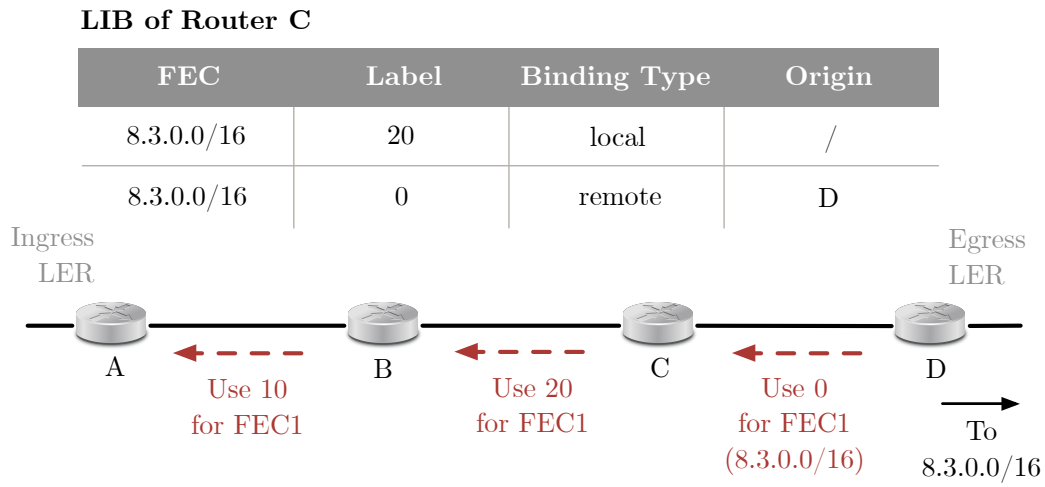


(b) Data Plane.

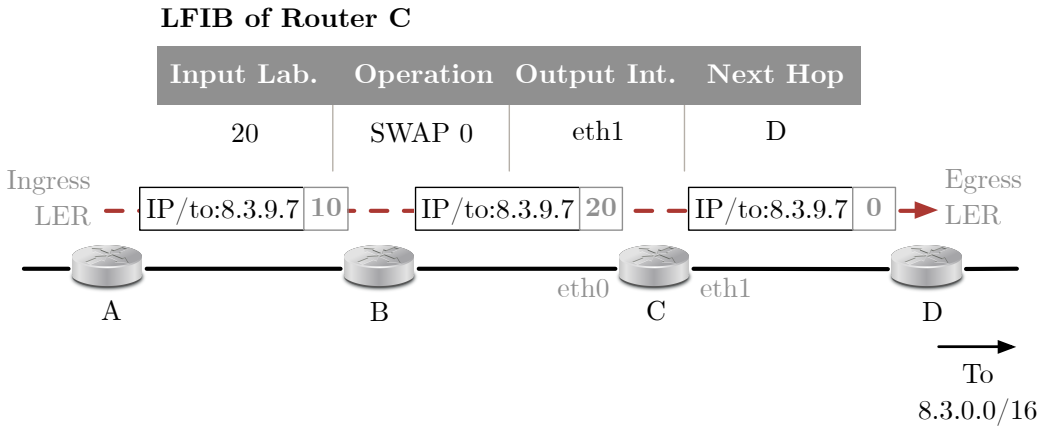
Figure 1.18: Penultimate Hop Popping (PHP).

class field, for example, the MPLS labels may need to reach the egress LER, and the Implicit NULL value cannot be used.

The reserved values 0 and 2 represent respectively the IPv4 and IPv6 *Explicit NULL Labels*, and are legal only at the bottom of the stack. They specify the MPLS router that the label stack must be popped, and that the packet must be forwarded based on its IP header. Similarly to the Implicit NULL Label, these values are announced by the egress LER. However, as they are explicit, they are written in the packet header. When used, the egress LER is ensured to receive a labelled packet, with all the information that would have been lost with PHP. This behavior is called *Ultimate Hop Popping (UHP)*, and is described in Figure 1.19.



(a) Control Plane.



(b) Data Plane.

Figure 1.19: Ultimate Hop Popping (UHP).

The last reserved value worthing to be mentioned is 1. It may be found anywhere in the label

stack, except at the bottom. It represents a *Router Alert Label*, and has a role which is similar to the *Router Alert Option* [84] in IP packets. It allows an LSR to intercept the labelled packet, even if it is not the destination. When a packet with label 1 on top of the stack is received, it is delivered to a software module for processing. The second value in the stack determines then the actual forwarding operations. If the packet must be transmitted further in the network, the LSR pushes back the Router Alert Label on top of the stack.

In practice, other label ranges than 0 to 15 can be reserved for specific use. For example, a block may be allocated for the *segment routing* technology [63]. This specific range may differ depending on the router manufacturer, and can be modified by network administrators. By default, CISCO devices use values between 16,000 and 23,999 [37].

1.6 MPLS Use Cases

MPLS was originally designed in order to increase the packet forwarding speed. Nowadays, its use is not limited to the improvement of network performance. The protocol is the main element in a variety of technologies deployed on the Internet.

1.6.1 Virtual Private Networks

One of the most popular fields of application of MPLS is the *Virtual Private Network* (*VPN*) [110, 128]. A VPN is a private network that interconnects remote sites over a public network, called *backbone*. In the VPN terminology, the owner of the different sites is the *customer*, while the operator of the backbone is the *service provider*.

The main advantage of VPNs is that they allow the different sites to exchange data securely, as if all devices were connected to the same network. An independent administration, with its own private IP addressing and routage can be applied over the whole virtual network, as the backbone seems invisible to the customer.

In this architecture, illustrated in Figure 1.20, *Customer Edges (CEs)* routers are directly connected to a *Provider Edge (PE)* router. Each CE device announces its IP prefixes to the PE gateway to which it is attached. The service provider uses then BGP to exchange the prefixes between the different customer' sites. Once data must be transferred, the backbone establishes a virtual connection between the two PE routers implied in the communication, which can be achieved thanks to the MPLS technology.

When an IP packet is received by a PE router, it is encapsulated with MPLS labels, and forwarded by the *Provider (P)* core equipment towards the exit node attached to the destination site. At this point, the label stack is popped and a simple IP datagram is sent to the CE router. In this structure, PE routers are LERs, while P routers are LSRs.

In a standard MPLS network, once the egress LER has removed the label stack, it must forward the packet based on its destination IP address. However, using private IP addressing

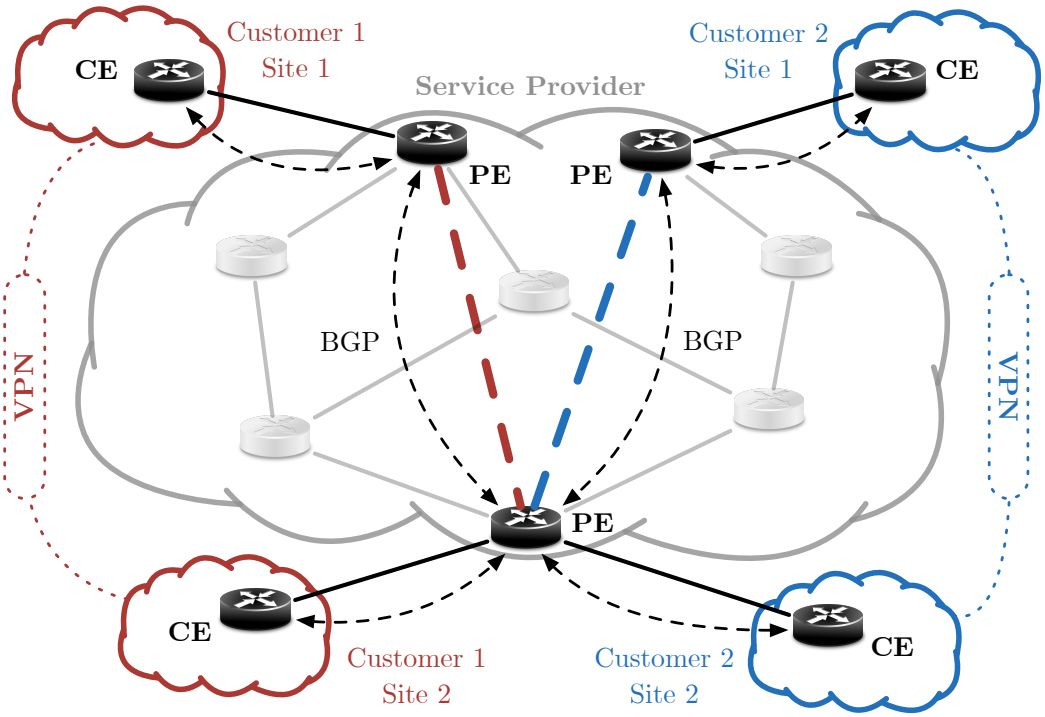


Figure 1.20: Virtual Private Network (VPN) architecture.

spaces is allowed in VPNs. A PE router may then be connected to different customer sites using overlapping IP address ranges. Forwarding the packet based on its destination is then impossible. The solution to this problem is to associate an identifier to each route within a VPN. It allows the network to identify correctly the two sites that are communicating. Based on its value, a border router is able to determine the right customer to which the packet must be forwarded. The identifiers chosen for each VPN route are distributed by BGP during the sessions established to exchange IP prefixes between sites.

In order to allow the PE router to transmit the data properly, the route identifier must be inserted in each packet at the entrance of the network. A two-layer MPLS label stack must then be used in the backbone, as shown in Figure 1.21. The bottom label identifies the VPN route, and must reach the egress LER. It is never modified by the intern al LSRs. The value on top of the stack identifies a FEC, and is used to forward the packet inside the provider's network. It is swapped at each hop in the domain until the egress LER is reached (or the penultimate hop if PHP is enabled).

1.6.2 Traffic Engineering

Today's Internet is an interconnection of organizations of different sizes having their own administration. Some of them are crossed everyday by a huge amount of data flows that may cause

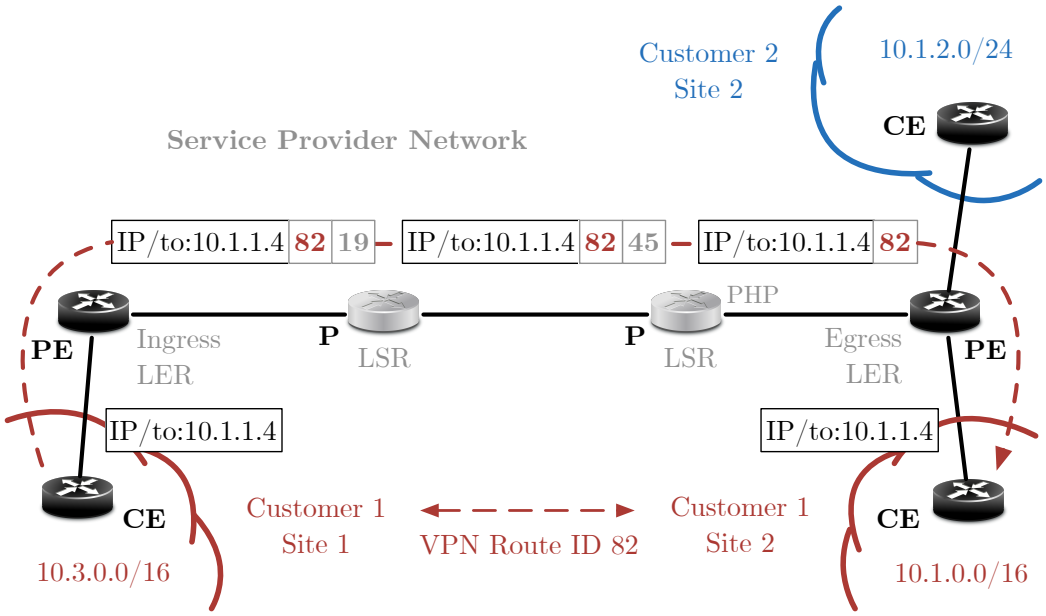


Figure 1.21: Data transfer in a Virtual Private Network (VPN). Two sites of Customer 1 exchange data. The VPN route identifier 82 is used as bottom label in the MPLS tunnel. The top label is swapped at each hop, and allows the LSRs to retrieve the forwarding instructions from their LFIB. The second internal LSR is in charge of popping the top label as PHP is enabled. Based on the bottom label value, the egress router knows the packet must be sent to Customer 1.

delay or packet losses in some parts of the network [113]. In such situations, a more flexible management of the traffic, such as redirecting some flows through a less congested path, or reserving resources for important connections, may help to improve the service quality.

Traffic Engineering (TE) [13] can be defined as a branch of network engineering that evaluates and optimizes the performances of operational IP networks. A TE solution may steer the traffic across the domain in order to efficiently use the available resources. As explained in Section 1.2, the forwarding decisions in an IP network are taken independently at each router, which makes difficult the deployment of traffic engineering solutions.

By its design, MPLS is a good candidate for network traffic optimization [15, 139]. With RSVP-TE, previously described in Section 1.5.4, the ingress LER is able to pre-compute a LSP which may be different from the path selected by IP. The protocol allows also to reserve resources along this route to ensure a quality of service to specific data flows. A network running RSVP-TE could dynamically modify its LSPs to improve its performances.

An operator may deploy simultaneously LDP and RSVP-TE in his network [2]. In this case, LDP is in charge of establishing MPLS paths following standard IP routes. Then, RSVP-TE can be used in order to compute new paths, or reserve resources depending on the current traffic and the network state. This practice is confirmed by a survey carried out as part of this thesis between August 28th and September 12th, 2017. In this survey, out of the fifty network operators

contacted, 87% declared having deployed MPLS in their network. Around 50% of them only use LDP to distribute labels, while 8% work only with RSVP-TE. As a result, both protocols are deployed in 42% of the cases.

A study of the deployment of MPLS traffic engineering solutions on today's Internet will be conducted in Chapter 4.

1.6.3 Fast Reroute

When a link or node failure occurs in an IP network, the different routers must signal the problem to each other, and re-run their routing algorithm in order to compute new routes avoiding the affected equipment. All these recovery operations may take a long time [66], during which the network is unable to continue forwarding the packets to their destination.

Fast Reroute [115] is another well-known field of application of MPLS. This mechanism, based on an extension of RSVP-TE, allows the creation of backup LSPs in order to protect an MPLS tunnel from a failure. It ensures that if a link or a node encounters a problem, the traffic is locally re-directed in a few 10s of milliseconds via an alternative route as close to the failure point as possible, as shown in Figure 1.22. The affected area is bypassed by the new path, repairing so the MPLS tunnel. The technique does not require any new route computation or failure notifications between LSRs, as the backup LSPs are announced in advance, before any problem may occur in the network. Once an LSR detects a problem, it deviates directly the data flow via its assigned backup path, if any. Fast Reroute allows to re-direct the traffic without interruption while other MPLS routers compute a new end-to-end LSP to replace the currently affected path. Resources may be reserved on the backup path in order to continue meeting a guaranteed quality of service.

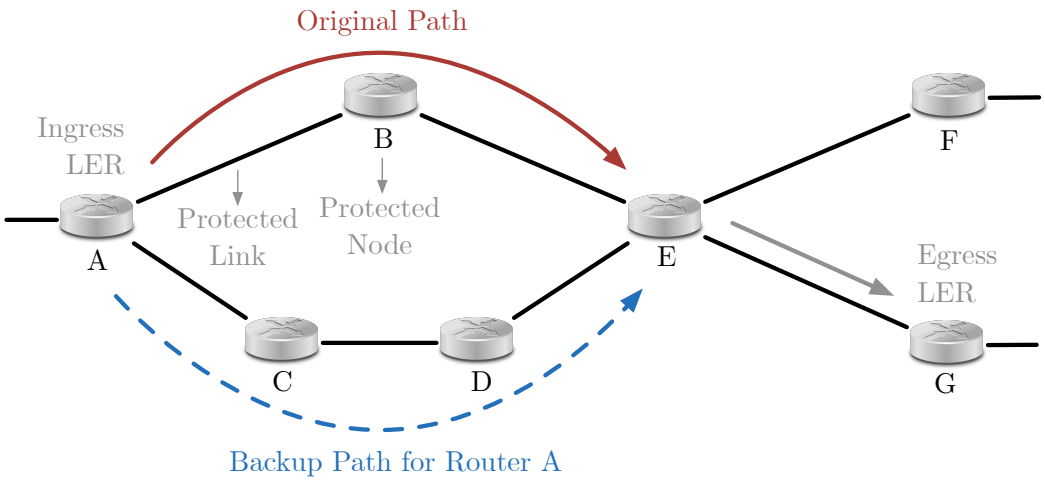


Figure 1.22: MPLS Fast Reroute. RSVP-TE announced the backup path A-C-D-E in case of failure at router B or at the link between A and B. In order to fully protect the original LSP, backup LSPs should be associated to routers B and E too.

1.6.4 Transit Traffic

Each time a packet is received at the entrance of an operational network, it must be forwarded based on its destination. If the receiver is located outside the domain, the packet is considered as part of a *transit traffic*. Most of the time, it is sent as fast as possible to an exit point, in order to ensure that it will consume as less resources as possible.

MPLS can be used to forward in-transit packets in a network. Two different architectures may be deployed by operators.

In the first solution, adopted by default in JUNIPER routers [16], MPLS tunnels are created for transit traffic only, while other data flows are still transferred with IP. Border routers may play the role of LERs, and establish LSPs between each other when needed, as illustrated in Figure 1.23. If a packet must cross the network, the border router determines the egress LER based on the destination, and pushes a label stack in the header. The internal LSRs forward this packet based on the label values only, without being aware of external IP prefixes. External destinations are then not injected in their forwarding tables to reduce their size. The complexity is transferred to the edge of the network, and the transit traffic is separated from other data flows, which eases the administration in the domain.

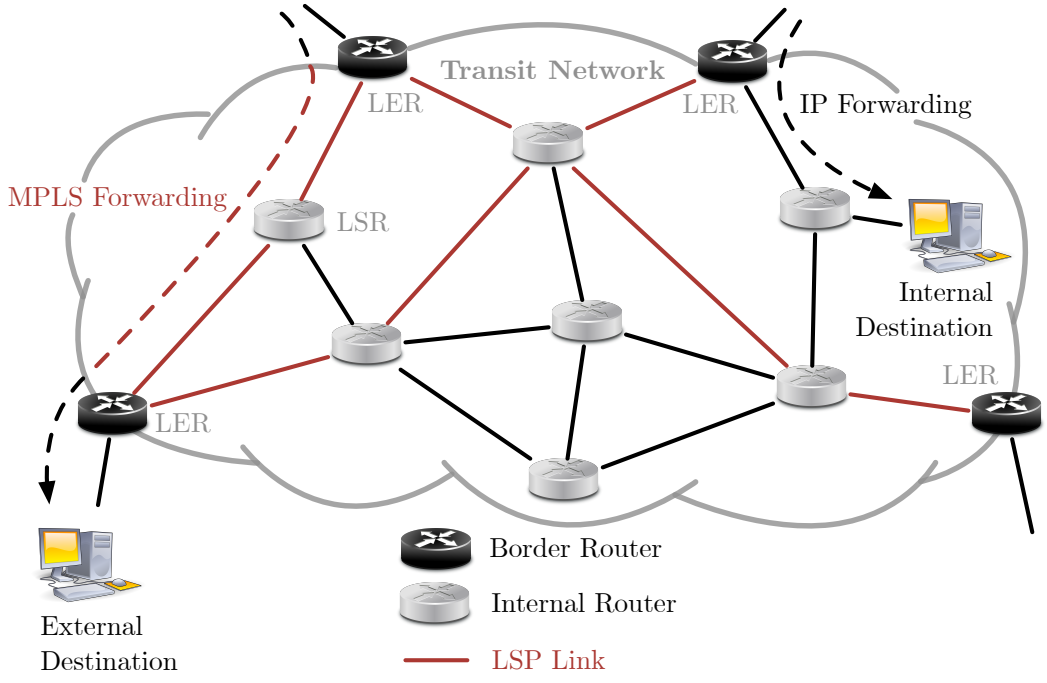


Figure 1.23: MPLS transit network. Border routers play the role of LERs and are interconnected thanks to LSPs. MPLS tunnels are only used for transit traffic. Packets with a destination belonging to the transit network are still forwarded with IP.

In the second architecture, implemented by default in CISCO routers [48, Chap. 4], MPLS

is also used for all internal prefixes, in addition to transit traffic. Tunnels are thus created for destinations inside the network too. Any internal router may then play the role of an LER, depending on the receiver of the data flow. In opposition to the first structure, no more separation exists between transit and other types of traffic, and the network is fully MPLS-capable.

The use of MPLS in transit networks is addressed in more detail in Chapter 4.

1.6.5 Transition from IPv4 to IPv6

Nowadays, the most deployed version of IP, *IPv4*, encodes the address of a network interface on 32 bits, which allows around 4 billions of possibilities. With the increasing number of devices being interconnected on the Internet today, this representation is no longer suitable. On September 24, 2015, the free pool of IPv4 address space of the *American Registry for Internet Numbers (ARIN)* [8] depleted, and since this date, the organisation is no longer able to allocate any new IPv4 block [5], except to facilitate the transition to *IPv6* [49].

IPv6 is another version of IP, designed in the 90s, in which addresses are represented with 128 bits. Even if it allows to solve the problem of the limited size of the IPv4 address space, the adoption of this protocol was quite slow until recent years [43].

One of the difficulties of migrating from IPv4 to IPv6, is to allow IPv6 networks to communicate on the Internet, where IPv4 is still the most present. As MPLS routers forward packets based on label values, they do not need to understand their actual destination. The protocol can then be used to allow IPv6 flows to transit in a network unable to perform IPv6 forwarding. The mechanism interconnecting IPv6 islands over an MPLS-enabled IPv4 cloud is called *6PE* [47], and will be addressed in detail in Chapter 5.

CHAPTER 2

MEASUREMENT-BASED CLASSIFICATION OF MPLS TUNNELS

The most popular tool allowing to infer the IP path followed by packets between a source and a destination is `traceroute` [58]. It is intensively used to collect measurement data in multiple research fields [52], as well as to identify possible issues in operational networks [1]. As explained in Chapter 1, MPLS is a protocol that forwards packets based on label values instead of destination IP addresses. It may influence the results obtained during `traceroute` measurements, and lead to erroneous interpretations, as the tool was originally designed to operate on native IP networks. This chapter presents a classification of MPLS tunnels based on how they react to `traceroute` and `ping` [111] probes, as introduced by Donnet et al. [53].

2.1 Introduction

Each IP packet generated by a device contains a TTL field in its header. It is used to prevent this packet from looping indefinitely in a network, and thus consuming to much resources. The value it contains is decremented at each hop on the path to the destination. When a packet with a TTL equal to 1 is received by a router, it is dropped, and an ICMP `time exceeded` message is sent back to the source to warn it of this operation. The functioning of `traceroute` is based on this behavior. The tool sends multiple probes towards a destination with an increasing TTL value, starting at 1. Each intermediate router on the route is forced to drop a packet, and is supposed to send back an ICMP message to to the source. These messages are used to infer the IP path, as they contain the IP address of the router interface from which they were sent. In practice, different types of probes can be used by `traceroute`. Each of them has advantages and disadvantages [98]. For example, UDP (*User Datagram Protocol* [119]) probes reach less destinations compared to others, but allow to infer more IP links, while TCP (*Transmission Control Protocol* [122]) ones succeed more often in crossing firewalls. On their side, ICMP probes hit more destinations, but may miss

some IP links in load-balanced paths. Choosing the right type of probes depends thus on the purpose of the measurements. Note also that a well-known modified version of `traceroute` has been implemented by the research community. This version, called `paris-traceroute` [10, 42], takes into account load balancers, and ensures that the different probes follow the same IP path towards the destination.

If `traceroute` is run to collect paths on the Internet, its different probes may cross an MPLS tunnel. As described in Section 1.5.2 of the previous chapter, MPLS uses its own TTL field, stored in a LSE. Only the LSE TTL at the top of the label stack is decremented inside the tunnel. The different LSRs never modify the IP header, including its TTL field. So, depending on the initialization of the LSE TTL at the ingress LER, the probes may never be dropped inside the tunnel, and thus, the source may not receive any ICMP `time exceeded` message from the internal MPLS routers. Moreover, when a monitor running `traceroute` receives an ICMP `time exceeded` message, it still needs to find a way to determine if this message was sent by an MPLS or a standard IP router.

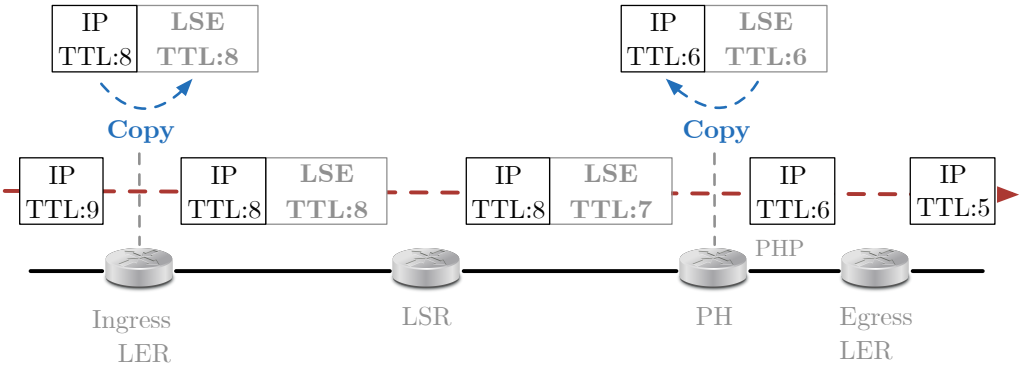
2.2 Measuring MPLS Tunnels

MPLS tunnels can be identified with `traceroute` based on two features implemented in the operating system of LSRs.

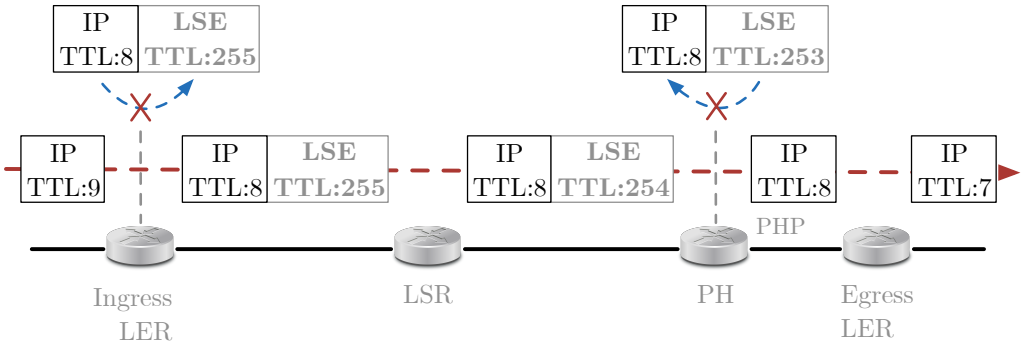
The first feature is an ICMP extension allowing network operators to verify their MPLS configuration. It is described in RFC4950 [21]. Similarly to IP, when the LSE TTL expires in an MPLS tunnel, an ICMP `time exceeded` message is sent back to the source. If the LSR implements the ICMP extension, it should quote the entire label stack of the dropped packet inside the ICMP response. Therefore, the presence of MPLS information in the message allows the source to determine if it was generated by an MPLS router. Modified versions of `traceroute`, such as `paris-traceroute`, already display in their output the LSE stacks received from LSRs.

The second feature is an option provided by router manufacturers called `ttl-propagate`. It is defined in RFC3443 [3]. It influences how the LSE TTL of a packet is initialized at the ingress LER, as illustrated in Figure 2.1. When enabled, the LSP is said being in *uniform* mode, and the IP TTL of the packet is copied in the LSE TTL field at the entrance of the tunnel. The opposite operation is also performed at the exit LER (or at the penultimate hop in case of PHP), just before popping the label stack. This option allows `traceroute` probes to expire inside MPLS tunnels. Consequently, internal LSRs send ICMP `time exceeded` messages back to the source. Unlike the ICMP extension defined previously, enabling the TTL propagation is the network operator's choice. If it is disabled (`no-ttl-propagate`), the LSE TTL is initialized to an arbitrary value, which is 255 in most cases. The LSP is then considered as being in *pipe* mode.

Only a few studies on the identification of MPLS tunnels with `traceroute` were conducted by the research community. Sommers et al. [137] evaluated the deployment of MPLS based



(a) Uniform mode (`ttl-propagate`). The IP TTL is copied in the LSE TTL field.



(b) Pipe mode (`no-ttl-propagate`). The LSE TTL is initialized to 255.

Figure 2.1: TTL propagation in tunnels enabling PHP.

on `paris-traceroute` data collected by CAIDA's Archipelago measurement infrastructure [38] between June 2008 and August 2011. Their analysis only focused on the tunnels perfectly visible in `traceroute` outputs, i.e., implementing the ICMP extension, and enabling the TTL propagation. They showed that operators tend to increase their use of MPLS across the years. They observed that 25% of the traces collected in 2011 crossed at least one MPLS tunnel. They identified tunnels in 7% of the observed ASs, with the largest deployment in Tier-1 networks. This proportion remained constant during the three years of measurements. They also developed a passive inference method using a Bayesian data fusion methodology in order to identify in their traces LSRs that do not include the label stack in their ICMP `time exceeded` responses. This method was based on round trip time measurements linked to the observation that when an internal MPLS router must generate an ICMP `time exceeded` message, the packet is first forwarded to the end of the tunnel prior to being sent back to the monitor (therefore, the delays should be relatively equivalent for the different LSRs along a tunnel). On their side, Sherwood et al. integrated into their Internet cartographers PASSENGER [136] and DISCARTE [135] a technique allowing to reveal hidden routers based on the IP Record Route option. They defined

hidden devices as routers that do not decrement the IP TTL, and thus, that may not appear in `traceroute` outputs. With DISCARTE, the authors discovered that 0.3% of the routers in their dataset collected from PlanetLab [32] were hidden from `traceroute`. However, they observed that MPLS devices do not write their IP address in the IP Record Route field. Therefore, tunnels that disable the TTL propagation could not be identified with their solution. Moreover, due to space limitation in the IP header, only 9 IP addresses can be written in the record route array, limiting so the scope of the technique. Marchetta et al. also focused on the identification of hidden routers using ICMP Parameter Problem messages [102]. These messages are generated by routers when they detect a problem that is not covered by other types of ICMP replies. The suggested methods use the IP Record Route and Timestamp options to force routing devices to send Parameter Problem replies. However, based on a PlanetLab campaign in September 2014, the authors highlighted that, on average, only about 60% of the routers observed on a path reply with an ICMP Parameter Problem message. Marchetta et al. also decided to directly exploit the ICMP Timestamp option in their tool DRAGO [103]. This last solution is able to locate hidden devices, but cannot retrieve any of their IP addresses. Moreover, the authors showed that only 60% of the routers in the core network support the Timestamp option. However, they could infer from their dataset collected from Napoli that 6% of the traces contain at least one hidden device.

This thesis is a complementary work to the different studies discussed above. It aims at updating the state of the deployment of MPLS tunnels, as well as evaluating their use by network operators, but also focuses on new mechanisms to identify and reveal hidden MPLS routers. These mechanisms do not suffer from the limitations of the existing solutions.

Depending on whether an operator disables the TTL propagation or/and uses an older equipment not implementing RFC4950, `traceroute` outputs may be incomplete. As a consequence, false links might be inferred, and the actual presence of MPLS may be underestimated. Consequently, the first step of this work is to take into account all possible combinations of the two features, as suggested by Donnet et al. [53].

2.3 Classification

Donnet et al. [53] were the first to create a complete practical taxonomy of MPLS tunnels based on the RFC4950 extension and the TTL propagation. It is illustrated in Figure 2.2. The remainder of this chapter describes the four tunnel types they have identified according to the combination of the two MPLS features, as well as the identification techniques they suggested to recognize three of their classes in `traceroute` data.

2.3.1 Explicit MPLS Tunnels

The *explicit* class gathers MPLS tunnels where the TTL propagation is enabled, and composed of LSRs implementing the extension described in RFC4950. As shown in Figure 2.2, they are

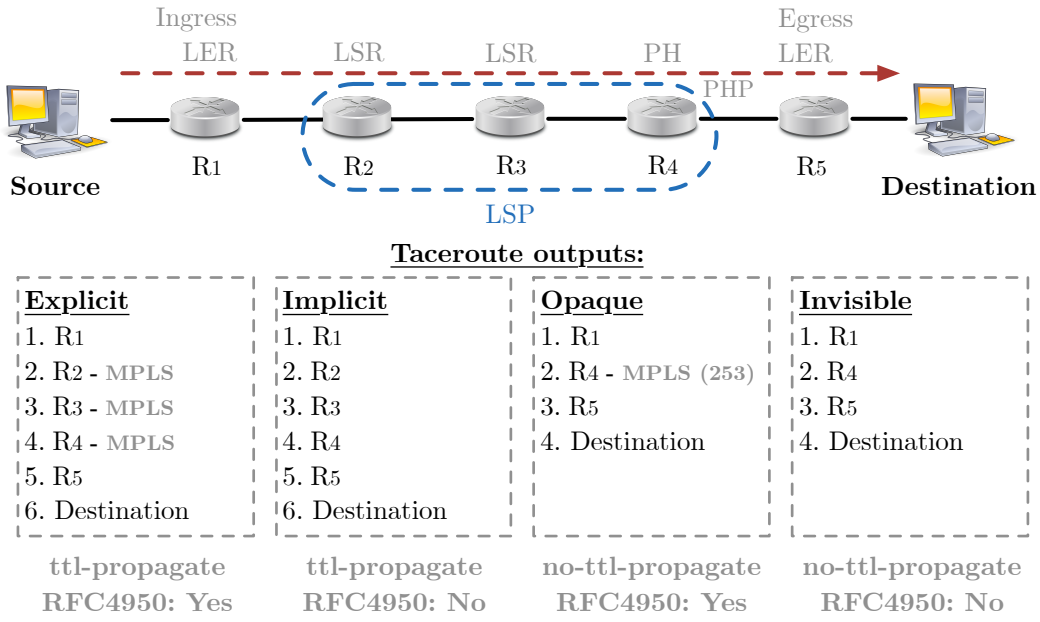


Figure 2.2: Taxonomy of MPLS tunnels as described by Donnet et al. [53], and their corresponding `traceroute` behaviors when PHP is enabled. Each class is defined in function of the implementation of RFC4950 and the activation of the TTL propagation.

perfectly visible in `traceroute` outputs. The label values used by the different MPLS routers can also be retrieved. This type of tunnel is the one on which multiple MPLS researches are based, such as, for example, the work of Sommers et al. [137], and the study of Al-Qudah et al. [4] highlighting that the protocol decreases the persistence and prevalence[1] of Internet paths, and so, their stability.

2.3.2 Implicit MPLS Tunnels

The LSRs of an *implicit* MPLS tunnel do not implement the ICMP extension. However, the TTL propagation is enabled. As a result, even if the different LSRs respond to `traceroute` probes, they do not include the label stack in their ICMP messages. The source is then not able to determine that they are MPLS routers, as shown in Figure 2.2.

Donnet et al. [53] present two inference methods to detect implicit tunnels in `traceroute` data. They are both illustrated in Figure 2.3.

The first technique is based on the IP TTL. As specified in RFC792, a router that generates an ICMP `time exceeded` message is required to include in the packet the IP header and the first 64 bits of the original datagram's data. As the IP TTL is part of the IP header, it is quoted in

[1]The authors define the persistence as "the amount of time it takes for an observed path between a source-destination pair to change", and the prevalence as an evaluation of "how often a certain path between two endpoints is taken".

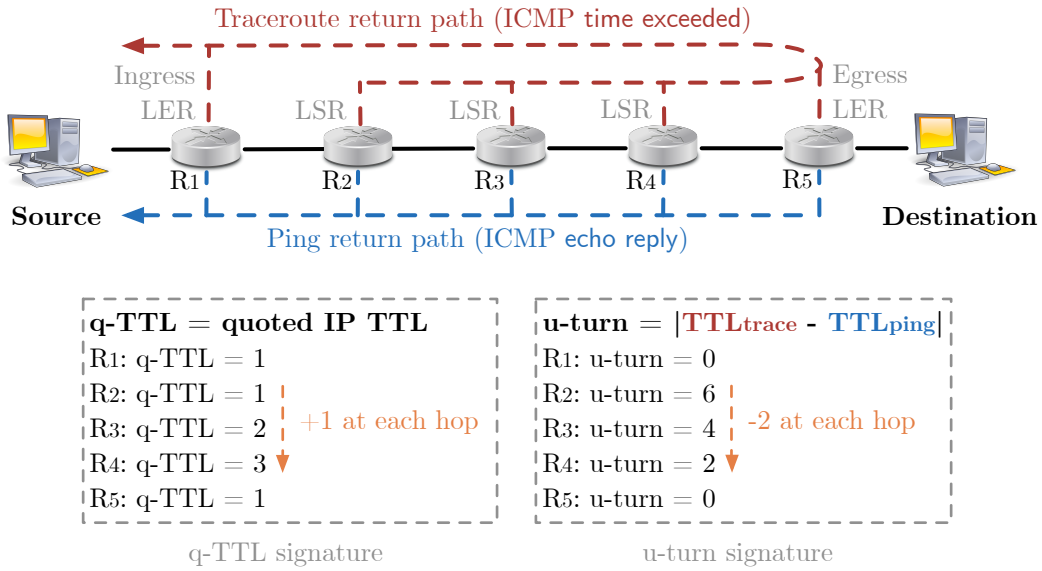


Figure 2.3: Inferring implicit MPLS tunnels, as described by Donnet et al. [53]. The q-TTL signature corresponds to increasing values of the IP TTL quoted in ICMP `time exceeded` messages. The u-turn signature highlights the difference of length between the paths used by `time exceeded` and `echo reply` messages. Theoretically, a decrease of 2 should be observed for each internal LSR.

the ICMP response. In an MPLS tunnel, internal routers do not modify the IP TTL field, which implies that the quoted value (q-TTL) increases for each new LSR responding to a `traceroute` probe. An increasing sequence of quoted TTL suggests then the presence of an implicit MPLS tunnel. This behavior, called q-TTL *signature*, can be observed in Figure 2.3.

The second inference technique is related to a particular behavior adopted by LSRs regarding ICMP *error messages*, such as `time exceeded` ones. When this kind of packet is generated in an MPLS tunnel, it is first forwarded (with MPLS) to the egress LER, which in turns transmits it to its destination[2]. ICMP *information* messages are not concerned by this specific practice. They are directly sent along the IP route to their destination, if a corresponding entry is available in the router's forwarding table. As a result, additional `ping` probes can be used to identify implicit tunnels. Similarly to `traceroute`, `ping` is a popular networking tool based on ICMP. It allows to test the reachability of a connected device. Its principle is quite simple, as it consists in sending ICMP `echo request` probes towards the target, which then replies using an ICMP `echo reply` packet. If a `ping` is run towards an LSR, the reply packet should follow a shorter path than the message used to respond to the `traceroute` probe. The difference between the IP TTLs of the `time exceeded` and `echo reply` pakets is then not null. Therefore, if this difference is computed

[2]This behavior is implemented by default in CISCO devices, and can be disabled using the command `mpls ip ttl-expiration pop <stack size>`. However, on the contrary, JUNIPER routers do not forward error messages to the egress LER by default. This operation must be enabled using the command `icmp-tunneling`.

for each MPLS router in the tunnel, a decreasing sequence should be observed until reaching the egress LER. A specific signature, called *u-turn signature*, can thus be identified. In the ideal case where the ingress LER is on the path from the egress LER towards the source, as illustrated in Figure 2.3, the signature is in the form *X, X-2, X-4, X-6, ..., 2, 0* where *X* represents two times the LSP length (i.e. its number of LSRs).

Both inference techniques have limitations in practice. For example, some LSRs may set the IP TTL with the LSE TTL value before sending the ICMP `time exceeded` message. If so, the q-TTL signature does not appear in the `traceroute` output. In some configurations, internal routers may directly send `time exceeded` messages to the source, without forwarding it to the egress LER. The IP route to the monitor may also be through the egress router. In both cases, the return paths of `time exceeded` and `echo reply` messages are similar, and the u-turn signature is not visible. Sometimes, LSRs in a tunnel may use different AS exit routers to send their `echo reply` packets, and the u-turn signature may differ from its ideal version, complicating its identification. Finally, network operators may filter ICMP `echo request` messages at the entrance of their networks, or may not inject external IP prefixes in the forwarding tables of their LSRs. In these situations, the MPLS routers are not able to answer to `ping` requests from the monitor. In practice, it has been demonstrated that the q-TTL signature is more reliable than the u-turn technique [44].

2.3.3 Opaque MPLS Tunnels

In *opaque* tunnels, the TTL propagation is disabled, but the LSRs follow RFC4950. The different internal LSRs are not visible in the `traceroute` output, except the penultimate hop if PHP is enabled, as shown in Figure 2.2. As the LSEs are included in the ICMP `time exceeded` response, the source is able to identify the MPLS node that popped the label stack, and the tunnel appears as containing only one LSR.

The length of opaque tunnels can be inferred based on the LSE TTL quoted in the ICMP message. As explained in Section 2.2, the ingress LER initializes its value to 255 as the TTL propagation is not enabled. In the tunnel, each internal LSR decrements the LSE TTL, so that it reaches *255 - n + 1* at the penultimate hop, *n* being the number of LSRs in the LSP. A simple computation can then derive *n* from the value received in the ICMP packet. For example, in Figure 2.2, the LSE TTL is set to 253 when the packets arrives at the PH, as it has been decremented at R_1 and R_2. Based on the previous formula, the LSP length is inferred as 3, as expected.

This model of opaque tunnels, suggested by Donnet et al. [53], is not fully correct. In reality, this kind of tunnel is related to a specific behavior of CISCO routers. This will be discussed in more detail in Chapters 3 and 7.

2.3.4 Invisible MPLS Tunnels

The last class contains *invisible* MPLS tunnels. In this type of tunnel, the IP TTL is not propagated, and RFC4950 is not implemented. The `traceroute` behavior is similar to opaque tunnels, except that the label stack is never sent to the source. As a consequence, the tunnel is completely obscured, and does not appear in the output, as shown in Figure 2.2.

Similarly to opaque tunnels, the model shown in Figure 2.2 is not perfect, and does not take into account the fact that two different behaviors can occur depending on whether PHP is enabled or not. Moreover, invisible tunnels may be composed of LSRs implementing RFC4950. As the biggest part of this thesis focuses on the revelation of invisible MPLS tunnels, the subject will be addressed in detail in Part 3.

PART II

MPLS ARCHITECTURE

CHAPTER 3

NETWORK FINGERPRINTING

Fingerprinting network equipment, i.e., determining its operating system and/or manufacturer, has many potential applications and benefits in network management and security. More generally, it allows to better understand network structures and administration. It may thus be very useful to gain more insight on the behavior and architecture of MPLS tunnels. Moreover, as it will be shown in Part 3 of this thesis, it can be used to guide new techniques for revealing hidden MPLS routers in `traceroute` data. This chapter focuses on the description of a simple fingerprinting mechanism based on the initial TTL values used by routers to reply to various probing messages [147]. The presented technique comes at a very low additional cost compared to standard active topology discovery measurements, and allows to obtain classes that are meaningful to distinguish router platforms. At the end of the chapter, the fingerprinting method is applied to MPLS tunnels to study the type of device that compose them, and their impact on the tunnel behaviors.

3.1 Introduction

Fingerprinting [89, 99] refers to the act of dividing network equipment into disjoint classes by analyzing messages sent by that equipment, usually in response to some form of active probing. Those classes may correspond, for instance, to router *Operating Systems (OSs)*, router brands, or router configurations. Providing such a fingerprinting is useful for several applications and studies. For example, in network management, if it depends on the router OSs, it may help to list the network nodes and find vulnerable hosts in terms of security and fault tolerance [99, 148]. It may also be used to identify which nodes have an abnormal behavior (e.g., delay, packets drop/modification, etc). In network topology discovery [52], it could find a suitable use to under-

stand how various types of equipment are interconnected. Indeed, obtaining the router level map of a network from `traceroute` data requires an additional intensive probing step: *alias resolution* [88]. Router fingerprinting may drastically speed up this step, since IP addresses belonging to different classes cannot be aliases, and so, do not require to be further probed for alias resolution [72]. Another interesting application is the analysis of IP networks to check if they are heterogenous in terms of hardware and software at different scales (e.g., temporal and structural to study respectively the evolution and the internal structure of autonomous systems). Indeed, an accurate fingerprinting technique may allow to distinguish the router OS among a given brand.

However, fingerprinting can be costly and possibly intrusive as it may require many probes [99]. In this case, it becomes a time consuming process using undue network resources. Moreover, too many probes towards a network node, or a subnet, could be seen as remote host scanning and, consequently, be filtered.

The new fingerprinting technique presented in this chapter is a companion to `traceroute`-like exploration. The method is simple[1], requires few additional probes to `traceroute` ones, but still allows to classify Internet routers based on their hardware and OS. It infers the initial TTL values [45, 132] used by routers when generating packets in response to different kinds of probe. This set of values is called *router signature*. Router signatures are meaningful for fingerprinting as the initial TTL varies not only between different router platforms, but also in function of the protocol and the message type (error versus standard replies for instance). Indeed, no specific default value has been standardized for the TTL field.[2]

In this work, a router signature is a n-tuple made of n initial TTL values. Those values are derived from the TTLs included in different types of probe replies. The number and the variety of probes give the actual value of n. Only 2 values are considered in this thesis. Longer signatures (i.e., $n > 2$) may provide a better distribution among router OSs. Indeed, with a better discrimination, the significant classes should be more meaningful. However, the results of a first and incomplete study [80] tend to show that increasing the size of the signature does significantly improve the classification[3]. Moreover, note that the OS market is known not to be uniformly shared. Thus, for an application such as providing a pre-partition to speed up an alias resolution process, the interest may be limited at the granularity of this partition.

Based on a large-scale measurement campaign, we will first demonstrate that the suggested router signatures are consistent among measurement points. Then, after providing some general distribution results, we will apply the fingerprinting method to MPLS tunnels. More specifically, we will try to associate router signatures to the different tunnel classes presented in Chapter 2,

[1] Note that most routers do not reply to "complex" scanning tools such as `nmap` [99].

[2] It is worth noting that RFC1700 recommends to use 64 as initial TTL value [123]. This is however not followed by most router manufacturers.

[3] This study was based on a large measurement campaign, but no validation was performed. The behavior of well-known OSs could have been analyzed in a virtual environment with GNS3 [70], for example. Moreover, not all possible kinds of reply messages were investigated.

and see if specific behaviors depend on router manufacturers.

3.2 Network Fingerprinting

This section introduces the fundamentals of the fingerprinting method. In order to obtain replies from most routers, the probing mechanism must only rely on their standard behavior, and remain as basic as possible.

In the IP header, represented in Figure 3.1, the *Time to Live (TTL)* field is used to allow the network to drop a packet if a routing loop occurs. This 8-bit field is set by the originating host/router to an *initial value* which is usually and nearly always a power of 2 in the list 32, 64, 128, and 255. The TTL is decremented by 1 at each intermediate node along the path followed by the packet. When its value is 1, the router determines that the packet has consumed a sufficient amount of resources in the network, drops it, and informs the source by sending back an ICMP `time exceeded` message.

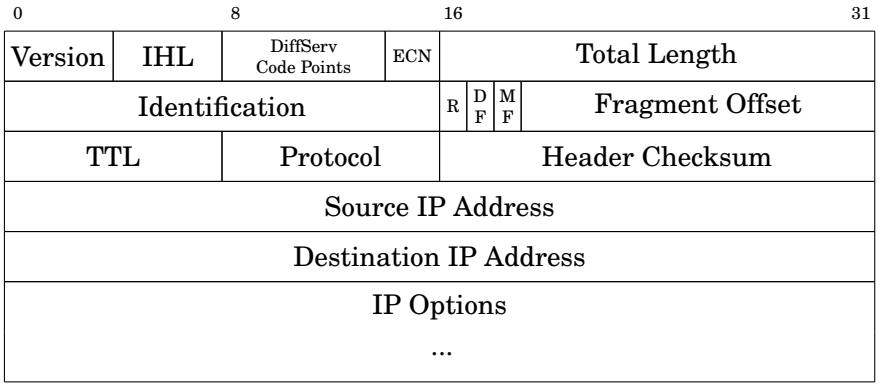


Figure 3.1: IPv4 header.

The fingerprinting mechanism, illustrated in Figure 3.2, requires to determine the initial TTL ($iTTL$) of the ICMP packets received by the source in response to its probes. Upon reception, the value in the TTL field is equal to $iTTL - \#hops$ where $\#hops$ is the number of hops between the sender and the receiver. Taking into account that most of the time, $\#hops < 30$ (99.8% of the paths in the dataset used for this study have less than 30 hops) [54], the iTTL value can be estimated as the smallest number in $\{32, 64, 128, 255\}$ that is larger than the received value. In very infrequent cases, an iTTL of 64 together with a very long route ($\#hops > 32$) would give an incorrect guess of 32 instead of 64. For example, a path of length 34 would be interpreted as a path of length 2. Those cases could be managed during a `traceroute` campaign by looking at the number of hops in the forwarding route. A difference between the number of forward and backward hops close to 32 would indicate that the iTTL is 64 instead of 32.

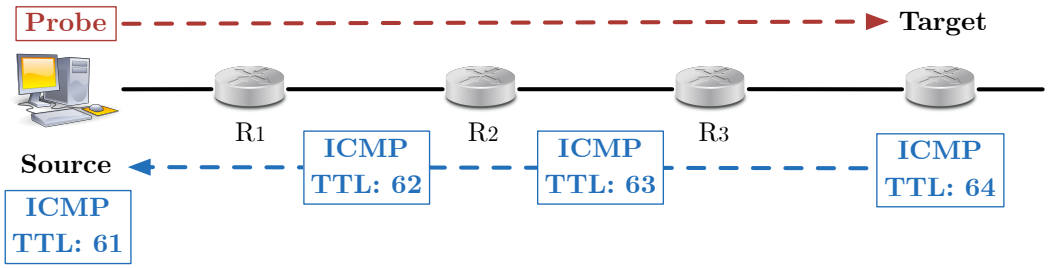


Figure 3.2: Illustration of the fingerprinting mechanism. A probe is sent towards the target which responds with an ICMP message. Once this message is received by the source, the initial TTL used by the target can be inferred as the smallest value in $\{32, 64, 128, 255\}$ larger that 61, which is 64.

A router *signature* is made of a n-tuple of iTTLs. These iTTLs are retrieved from different ICMP responses. The suggested basic signature simply uses two different messages ($n = 2$): a `time exceeded` message elicited by a `traceroute` probe, and an `echo reply` message obtained thanks to an `echo request` probe. It has thus the form $< iTTL_{TE}, iTTL_{ER} >$, where $iTTL_{TE}$ and $iTTL_{ER}$ represent respectively the initial TTLs of the `time exceeded` and `echo reply` responses. Quite surprisingly, a significant proportion of nodes use two different iTTL values for these two ICMP messages, as shown in Figure 3.3.[4] Note that adding a third iTTL to the signatures was considered. It would have been obtained based on a `destination unreachable` message elicited by a UDP probe. Figure. 3.3 shows that 40% of routers do not respond to such probes. Consequently, the basic signatures would have been extended at most with an absence of response. As this iTTL requires more probing, and brings limited information for the fingerprinting, it was not taken into account in this work.

In theory, using n probes may generate up to $4 \times 5^{n-1}$ different signatures, since a lack of response to some probes (i.e., a $*$) is a valuable pattern. Note that the "4" before the multiplication sign is due to the fact that `traceroute` is the basic probing mechanism, i.e., `time exceeded` messages are used to direct subsequent probes (`echo request`, UDP packets, ...). So, the non-response $*$ is not taken into account for these messages, and the only 4 possible iTTL values are 255, 128, 64, and 32.

3.2.1 Measurements

During the measurement campaign, IP level paths were collected thanks to `paris-traceroute` with ICMP `echo request` packets. Each received ICMP `time exceeded` message was used to generate the first component of the router signatures. In addition, for each discovered IP interface, six ICMP `echo request` probes were sent. The use of six messages ensured the robust meaning of a non-response $*$, and helped identify implicit MPLS tunnels (see Section 3.3). The different

[4]The methodology of the measurement campaign will be given in Section 3.2.1.

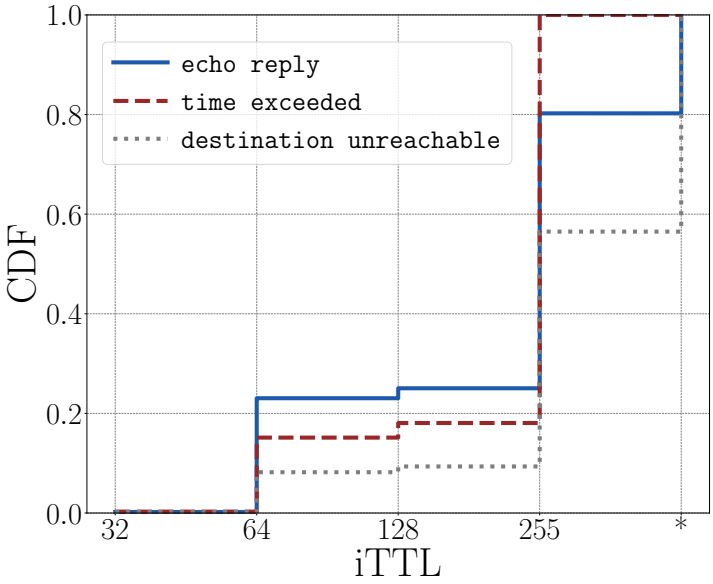


Figure 3.3: Distribution of initial TTL values ($*$ refers to non-responding routers).

ICMP `echo reply` packets sent by the routers allowed to complete the second component of the signatures. To limit the probing cost, each discovered IP address was pinged six times only once per vantage point (i.e., when it was discovered for the first time).

The measurements were performed by a team of 200 randomly selected PlanetLab [32] *Vantage Points (VPs)*. Out of the 200 VPs, 121 were located within the US, 10 were in Europe, and the other 69 in different countries. One million destinations were randomly selected in the Archipelago [38] target list and evenly distributed among the VP teams. The dataset was collected between January 8$^{\text{th}}$, 2013 and January 10$^{\text{th}}$, 2013 using `scamper` [96]. A total of 335,646 distinct addresses were discovered. After the measurements, each address was considered only once, and associated to its signature.[5]

3.2.2 Measurement Cost

The additional overhead to fingerprint a set of routers discovered with a `traceroute` measurement campaign comes at a low cost. For each identified IP address, k ICMP `echo request` messages must be sent, k being a robustness parameter. Usually, many traces sent by a VP reveal the same addresses that must to be probed only once [54]. So, the fingerprinting technique needs at worst k more messages per IP address than `traceroute`, and most of the time, much less. In the measurement campaign of this chapter (where $k = 6$), respectively 13,437,896 and 14,803,614 `time exceeded` and `echo reply` responses were collected. Therefore, on average, a probed router must send about the same number of `time exceeded` and `echo reply` messages.

[5]The fingerprinting technique was later integrated into `scamper` by Florian Hoebreck [55].

This number could even be further reduced by setting a smaller robustness factor, or by adding some extra cooperation between the VPs in order to avoid as much as possible pinging the same IP address several times.

3.2.3 Signatures Consistency

The objective behind fingerprinting is to obtain a signature that depends only on the probed node. To prove this hypothesis, the signatures of the IP addresses observed by at least two distinct VPs were compared. These signatures were thus obtained with distinct `traceroute` and `ping` probes. Note that only IP addresses discovered with `traceroute` probes were considered, but some of them did not always respond to `ping` requests (i.e., the second component of the signature may be $*$). The signatures associated to these addresses were classified into three categories:

- *coherent*: the same signature is observed for a given router interface among all VPs. In the collected dataset, 95.92% of the signatures are coherent. They represent the perfect case.

- *weakly incoherent*: for some VPs, the signature is of the type $< x, y >$, while it is in the form $< x, * >$ for others. This type of signature corresponds to 3.94% of the dataset.

- *incoherent*: different signatures are observed for a given router interface among the measurements done by all VPs. They constitute the worst case, but are also very infrequent (0.14% of the dataset).

Weakly incoherent signatures appear when monitors do not receive any response to `ping` requests at some time. The cause may be an overloading of the network, a rate limiting at the targeted node, or a filtering performed by the network operator, inducing the lost of the `echo request` or `echo reply` packets on some paths [83]. As most of the signatures observed in the dataset are coherent, keeping only the best version of a weekly coherent signature (meaning the complete version) seems legitimate, and was performed in this study. After this operation, around 17.5% of the entire set of IP addresses collected during the campaign were still associated to an incomplete signature of the type $< x, * >$. Most of them (i.e., 10.19%) were observed from several VPs, while the remaining (i.e., 7.31%) were only seen by a single monitor. In this last case, the number of incomplete signatures could have been further reduced by trying to `ping` the corresponding IP addresses from other VPs.

Due to their extremely low proportion, incoherent signatures may be explained by some artifacts. A possible cause may be *middleboxes* rewriting the TTL field [50, 87, 106]. Then, the same IP address may correspond to different nodes depending on the network location (*anycast* address). Finally, the previously mentioned ambiguity between the iTTL values 32 and 64 may also generate different signatures for the same router interface.

3.2.4 Signatures Distribution

The signatures of some well known manufacturers and operating systems could be verified based on data sheets and a bunch of tests in an emulation lab. They are available in Table 3.1. For instance, CISCO routers generate the signature $< 255, 255 >$, while JUNIPER ones produce $< 255, 64 >$ when running Junos, and $< 128, 128 >$ with JunosE. In addition, some BROCADE and ALCATEL equipment, and some LINUX boxes are associated to $< 64, 64 >$. Although these signatures encompass the main router vendors, different other platforms may behave similarly when fingerprinted, and use the same iTTLs. One could envision to extend the signature with other types of messages, or other criterions, such as the size of the ICMP packet, for example. However, as already mentioned in Section 3.1, a preliminary study shows that increasing the length of the signature (i.e., a n-tuple with $n > 2$) does not improve significantly the granularity of the classification.

Router Signature	**Router Brand and OS**
$< 255, 255 >$	CISCO (IOS, IOS XR)
$< 255, 64 >$	JUNIPER (Junos)
$< 128, 128 >$	JUNIPER (JunosE)
$< 64, 64 >$	BROCADE, ALCATEL, LINUX

Table 3.1: Summary of main router signatures.

Figure 3.4 illustrates the distribution of the main router signatures. The class $< 255, 255 >$, that includes CISCO devices, is largely dominant, and is associated to more than 50% of the IP addresses. The signature $< 255, 64 >$, representing JUNIPER routers running JunOS, and $< 64, 64 >$, gathering several vendors and OSs (including LINUX), correspond to about 11% of the cases each. The second most frequent class, $< 255, * >$, accounting for about 15% of the data, corresponds to an incomplete signature, and is probably mostly made of routers belonging actually to $< 255, 255 >$ or $< 255, 64 >$ that did not respond to `ping` for various reasons [83]. Finally, the class $< 128, 128 >$, including JUNIPER platforms running the JunosE system, represents around 3% of the IP addresses, while the remaining signatures are either incomplete or very rare. Therefore, at a global scale, the different results seem to reflect the actual market distribution.

To summarize, among different brands and OSs, routing equipments may initialize the TTL differently, but a single device can also use different iTTL values depending on the situation (at least, for Juniper routers).

3.3 Application to MPLS Tunnels

This section focuses on the application of the fingerprinting method to the different classes of MPLS tunnel introduced in Chapter 2. We will see if the type of equipment used in MPLS

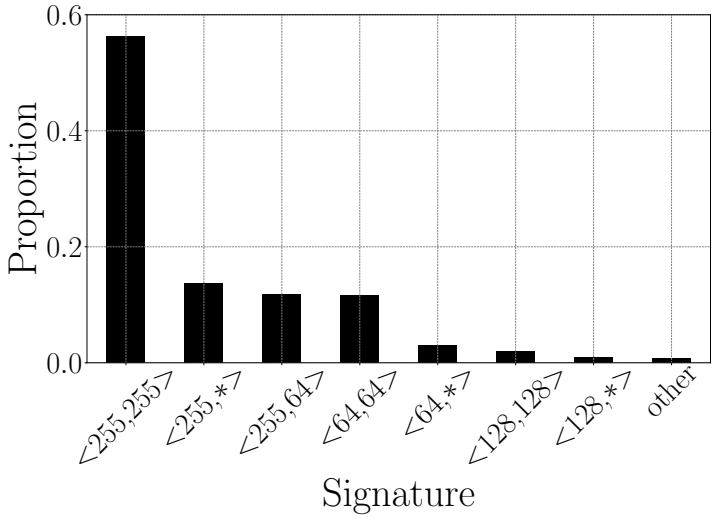


Figure 3.4: Distribution of the different iTTL signatures.

and IP networks is similar. We will also improve our understanding of opaque and invisible MPLS tunnels, and show that fingerprinting allows to determine whether a feature (here MPLS characteristics) is independent of the router type. More specifically, we will check if any correlation may exist between RFC4950, the `ttl-propagate` option, and the router platform.

3.3.1 MPLS Deployment

Generally speaking, Figure 3.5 shows that a large proportion of the paths collected during the campaign hits one or more MPLS tunnels. From about half of the VPs, at least half of the paths contains MPLS tunnels.

Explicit and implicit tunnels are by far the most frequent. Their number reaches respectively 46,657 and 61,054, which corresponds to roughly 40% and 60% of the IP addresses belonging to tunnels and, altogether, 17% of the collected IP addresses. However, most implicit tunnels are discovered using a probing inference heuristic (u-turn) whose accuracy is questionable [44]. Besides, opaque tunnels are rather rare[6], and so subject to weak statistics. Only 523 of them were observed. Finally, invisible tunnels could not be revealed, and their deployment was thus not evaluated.

3.3.2 TTL-Classification in MPLS Tunnels

Figure 3.6 highlights the signature frequency differences between MPLS (i.e., tagged as such by `traceroute` – implicit tunnels not included) and non-MPLS IP addresses. The first striking observation is that the signature $< 64, 64 >$ is much less prevalent in MPLS visible networks. A possible justification is that this signature may correspond to a variety of middleboxes, and

[6]This observation was expected as opaque tunnels result from a specific CISCO behavior (see Chapter 7).

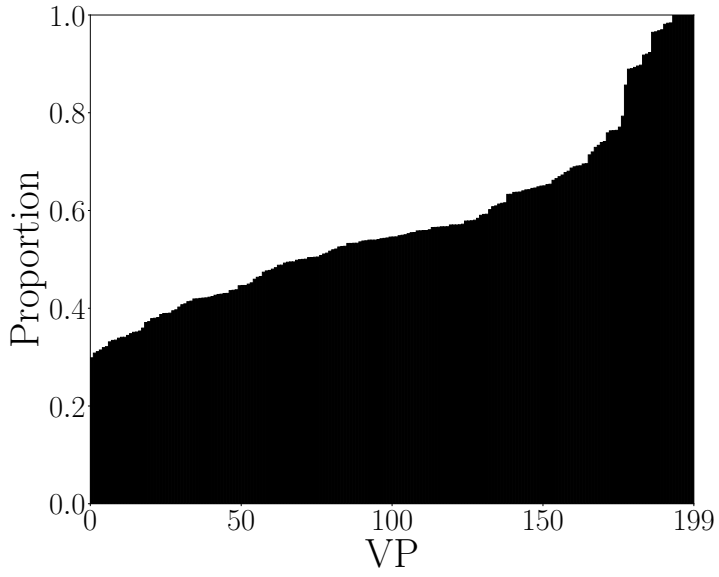


Figure 3.5: Proportion of paths, per VP, having at least one MPLS tunnel.

probably less to high-end routers commonly used in core networks, where MPLS is mainly deployed.

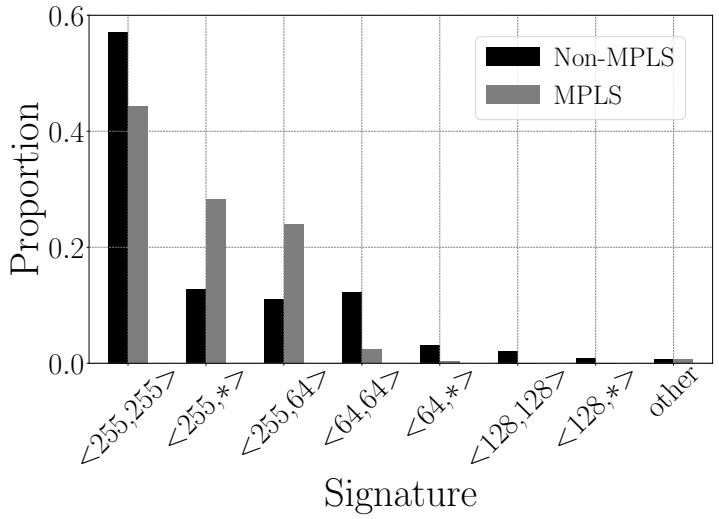


Figure 3.6: Signature distribution among MPLS and non-MPLS routers.

The second lesson is that the signature $< 255, 255 >$ is less dominant in MPLS networks, while $< 255, 64 >$ and $< 255, * >$ increase their share. The higher proportion of $< 255, * >$ may be caused by LSRs that do not have a complete IP routing table, and thus that cannot reply to `ping` requests (`time exceeded` messages are sent to the egress LER, and then forwarded to the source, as previously explained in Chapter 2). On its side, the signature $< 255, 64 >$ balances

probably the decrease of $< 255, 255 >$. Indeed, apparently, $< 255, 64 >$ routers (e.g., JUNIPER ones) increase their market position for MPLS operations (compared to the previous ratio $< 255, 255 >$ / $< 255, 64 >$ at a global scale).

Each of the MPLS tunnel classes described in Chapter 2 exhibits a specific router signature distribution, as presented in Figure 3.7. The X-axis gives the various signatures, while the Y-axis shows the proportion of tunnels, in a given MPLS class, that exhibits the specified signature.

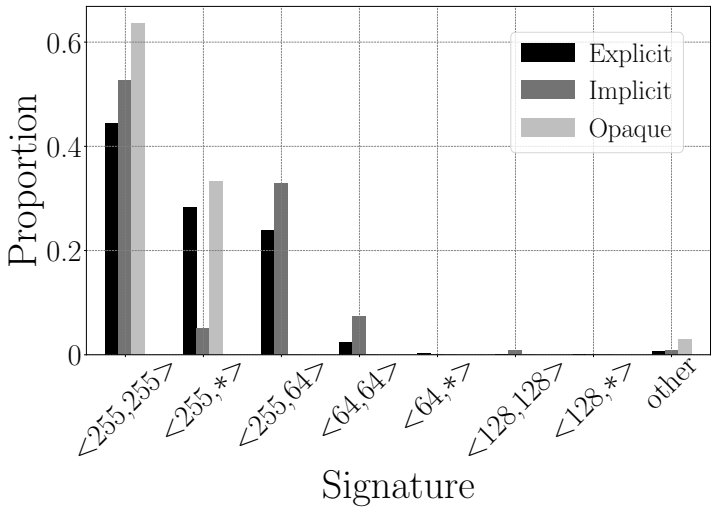


Figure 3.7: Router signature distribution among MPLS tunnel classes.

In the figure, opaque tunnels are only characterized by the signatures $< 255, 255 >$ and $< 255, * >$. Donnet et al. [53] assumed that a PH router would always insert a label stack in the ICMP `time exceeded` message if it implements RFC4950. Consequently, the difference between an opaque and an invisible tunnel was based on the RFC4950 implementation at the PH router. The new results clearly demonstrate that it is not the case. Indeed, if the PH does not belong to the class $< 255, 255 >$ (or its incomplete companion $< 255, * >$), it does not insert a label stack in the ICMP message, even if it implements RFC4950. The tunnel becomes then invisible. This assumption was verified with a virtual testbed. A tunnel made of LSRs running an RFC4950-compliant JUNIPER OS, and not propagating the IP TTL, could not be revealed. It was invisible, and not opaque. The number of invisible tunnels is then much more common in practice than stated by Donnet et al. [53], which will be confirmed in the third part of this thesis. Note that the case of opaque tunnels will be studied in more detail in Chapter 7.

A last question deserves attention: the amount of implicit tunnels being larger than the explicit one, are their detection mechanisms reliable? In Figure 3.7, the proportion of the signature $< 255, * >$ is less important in implicit tunnels than in other categories. The reason is obvious: u-turn signatures are prevalent in the tunnel detection, and cannot result from such a $< 255, * >$ pattern (`ping` replies are mandatory). The observed $\approx 5\%$ come from the q-TTL technique that does not depend on a probing heuristic. The proportion of $< 64, 64 >$ is also a bit higher (routers

that do no implement RFC4950 are likely to be older devices).

Except for these specific signatures, the distributions for implicit and explicit tunnels are relatively close. Such results tend to show that the detection of implicit tunnels seems quite reliable. However, the slight divergences are probably due to u-turn signatures. Indeed, Davila et al. [44] have shown that they may be biased, especially if a load balancer is present on the return path (i.e, the path followed by ICMP replies towards the source). They are thus more subject to false positives than q-TTL ones.

Figure 3.8 focuses on the implicit tunnel signatures to distinguish the two detection mechanisms. The signature $< 255, * >$ only exists with the q-TTL technique. The relative populations of $< 255, 64 >$ and $< 255, 255 >$ balance this decrease for u-turn tunnels in the same proportion as for MPLS explicit tunnels. On its side, the class $< 64, 64 >$ remains stable for both techniques. Hence, the u-turn mechanism is not the cause of the previous difference of $< 64, 64 >$ with explicit tunnels (that seems to be induced by the RFC4950 implementation).

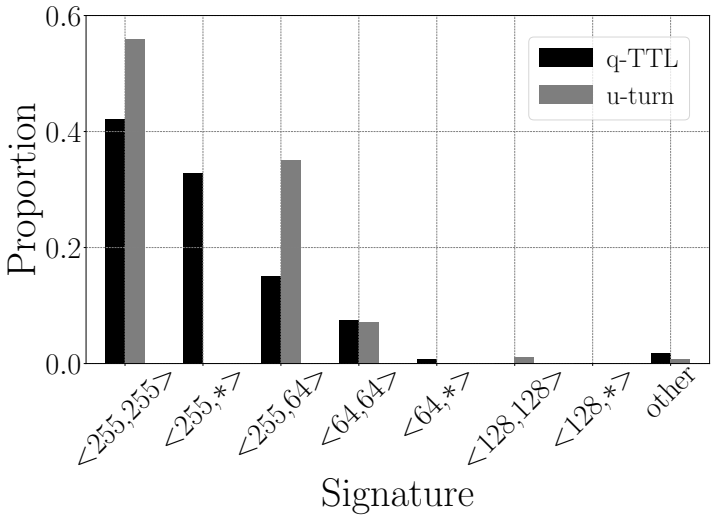


Figure 3.8: Router signature distribution among MPLS implicit tunnels.

3.4 Conclusion

Router fingerprinting may be helpful in many domains, such as alias resolution, and the detection of vulnerable routers or abnormal behaviors. This chapter focused on the presentation of a lightweight fingerprinting mechanism. It is based on router signatures, i.e., n-tuples made of initial TTL values used by a router when forging ICMP reply packets. Such signatures are suitable to consistently discriminate IP interfaces, as various manufacturers and OSs use different deterministic initial TTLs depending on the type of packet to generate. More generally, the presented method allows to determine if a sample of routers is representative of the Internet router mix, and check whether a routing feature is independent of the router type.

Based on data collected during a large-scale measurement campaign, the router OS distribution on the Internet was analyzed according to the signatures. MPLS tunnels were also fingerprinted to observe the equipment used for this technology, and validate heuristics allowing to reveal non-explicit MPLS tunnels. At the same time, the conclusions of Donnet et al. [53] about invisible and opaque tunnels were refined.

Finally, in the third part of this thesis, we will see that, in their default configuration, CISCO and JUNIPER devices behave differently regarding to MPLS. The fingerprinting technique will therefore be useful to guide the techniques allowing to reveal hidden MPLS routers.

CHAPTER 4

MPLS AND TRANSIT PATH DIVERSITY

Traffic engineering (TE) is one of the keys to improve packet forwarding on the Internet. It allows IP network operators to finely tune the paths followed by the data according to various customer needs. One of the most popular tools available today for optimizing the use of network resources in transit networks is MPLS. A *transit network* can be defined as a network that forwards traffic for which it is neither the source, nor the destination. On the one hand, MPLS and label distribution mechanisms such as RSVP-TE can be deployed in conjunction with BGP to define, for a given pair of edge routers, multiple transit paths verifying different constraints in a network. On the other hand, operators may simply enable LDP to distribute MPLS labels and improve the scalability of their network. In this case, all packets follow the same IP route between a pair of border routers. However, LDP may also generate path diversity, as it preserves the *Equal-Cost Multi-Path (ECMP)* feature of IGP routing, if any. In this chapter, using an MPLS label analysis, we will demonstrate that it is possible to better understand the transit path diversity deployed within a given ISP. More specifically, we will focus on the *Label Pattern Recognition (LPR)* algorithm [144]. Based on `traceroute` data including MPLS information, LPR is able to reveal the actual use of MPLS according to the inferred label distribution protocol, and to distinguish ECMP from TE multi-path forwarding.

4.1 Introduction

One of the cornerstones of the Internet is the way data is forwarded through routing paths. Typically, most of the IP flows are treated the same way, whatever their specific QoS needs, their destination, or their origin. This absence of privileges and flow distinction is called *best effort routing* or *Internet neutrality*. Tools allowing operators to easily enable path diversity, and so, to perform traffic engineering, are ECMP load balancers [11] at the IP level, and MPLS. Only a few

researches focused on the deployment of MPLS-based TE solutions [153] and their impact on the IP destination-based packet forwarding [64] and end-to-end delay [117]. However, none of them evaluated how MPLS is actually used on today's Internet.

Two label distribution protocols allow to build MPLS tunnels, according to their intended use. On the one hand, LDP[1] on top of IGP enables inter-domain routing stability and extensibility, but also preserves ECMP features of the underlying routing protocol, if any. On the other hand, RSVP-TE allows operators to enable service differentiation (i.e. TE) through the use of multiple FECs[2]. In practice, the same ISP may run concurrently both types of label distribution protocols depending on the tunnel use. Usually, LDP seems to be deployed by default (to build a full-mesh between edge routers) in MPLS networks, as it will be confirmed further in this chapter. Note that VPNs based on MPLS may rely either on LDP or RSVP-TE depending on the QoS requirements. However, they will not be considered in the following, as only a few tunnels through VPNs were observed in the measurement dataset collected for this study.

This chapter presents the *Label Pattern Recognition (LPR)* algorithm, allowing to differentiate standard IP equal-cost paths (that LDP allows) from transit MPLS tunnels built with RSVP for actual TE purposes. LPR is a passive algorithm, in the sense that it does not require any additional probing to standard `traceroute`. It must be run once the data has been collected, as long as this data contains the information related to MPLS tunnels, meaning the stack of LSEs returned in ICMP `time exceeded` messages. LPR focuses thus on *explicit* MPLS tunnels[3], implementing RFC4950, and propagating the IP TTL. Briefly, it classifies each tunnel <*entry point, exit point*> pair into one particular class according to the recognition of the standard behaviors of RSVP-TE versus LDP in terms of label distribution.

After a complete description of the new algorithm, we will study its application on an extensive dataset obtained from CAIDA's Archipelago measurement platform [38]. More specifically, we will analyze the use of MPLS in major autonomous systems on the Internet based on five years of `traceroute` data. We will observe each AS independently, and understand whether it enables path diversity, how, when, and if it evolved over time (e.g., from almost no path diversity to a wide deployment of TE).

4.2 Transit MPLS Tunnels

LPR allows to determine the use of transit MPLS tunnels. Two common scenarios may be observed in operational networks.

[1] LDP and RSVP-TE are described in detail in Section 1.5.4.

[2] A FEC refers to a set of packets a single router forwards to the same next hop, via the same interface, and with the same treatment, as described in Section 1.3.

[3] Explicit MPLS tunnels were presented in Section 2.3.1.

4.2.1 Basic Encapsulation with LDP

Using BGP as an inter-domain routing protocol and an IGP (e.g., IS-IS or OSPF) as an intra-domain routing protocol, a network operator may use MPLS tunnels between its border routers to transparently carry packets between them, as shown in Figure 4.1. This way, the intermediate devices do not need to know external destinations[4], only the incoming border routers must be aware of them (i.e. they must know the BGP next hops) and their corresponding LSP. Such an MPLS use may prevent routing loops and other kinds of anomalies [149], and mostly enables scalability. Another similar use is for basic BGP MPLS VPNs. Again, LSPs are established between the border routers of the provider such that packets belonging to a VPN may transparently cross the MPLS domain. In both cases, the objective pursued is to separate routing within the domain from routing outside (inter-domain routing or routing in the VPN), not to select routes in the network.

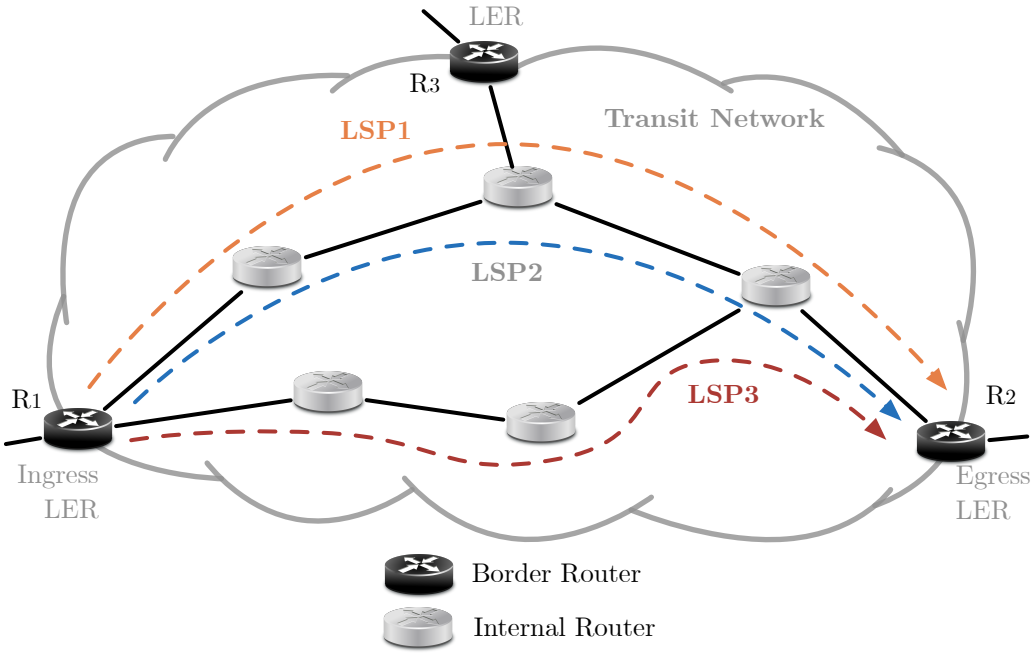


Figure 4.1: General overview of a network using MPLS for transit traffic. Three different LSPs are established between the ingress LER R_1 and the egress LER R_2. LSPs between an Ingress LER and an egress LER might be physically different (i.e., different IP addresses, as LSP1 and LSP3), or logically different (i.e., same IP addresses but different labels, as LSP1 and LSP2).

For this basic encapsulation method, labels are allocated through LDP. A router announces to its MPLS neighbors the association between a prefix in its routing table and a label it has chosen.

[4]It has been shown [53] that, in practice, a significant proportion of routers are unable to reply to `ping` if they are involved in an LSP (without being LER). It means that many LSRs do not have a global IP routing plan (no BGP redistribution within the IGP). Instead, IGP routers may use a default path obtained from a route server, for example.

Therefore, labels are allocated from downstream and, for a given prefix, a router advertises the same label to all its neighbors. Depending on the implementation, LDP may announce a label for all the prefixes in the IGP routing table (default case for CISCO routers [48, Chap. 4]) or only for loopback addresses (default case for JUNIPER routers [16]). For transit traffic, LSPs are constructed by LDP towards loopback addresses of the exit border routers (the loopback addresses become then the BGP next hops). The IP route followed by the LSP is the best effort IP route computed by the IGP. If no IGP load balancing is performed in the network, only one route exists between two endpoints (for instance, only LSP3 in Figure 4.1 between the ingress and the egress LERs). On the other hand, if load balancing is used, several routes may be observed (usually having an equal cost (ECMP) – for instance LSP1 and LSP3 in Figure 4.1). Load balancing is generally performed using a hash function on particular IP and transport header fields. In practice, LDP builds an LSP-tree towards the destination prefix. While the prefix used to build this tree may be very specific (i.e. a single IP loopback address), the FEC may be very large (e.g., all traffic exiting a Tier-1 AS from the same border router). Note that a FEC may be defined at a finer grain using access control lists, for instance. In this case, other fields that the IP destination can be considered.

4.2.2 Traffic Engineering with RSVP-TE

Another quite different use of MPLS is traffic engineering, where the goal is to tune the routes used by flows either to give them requested QoS, or to optimize the network use. In this case, different flows entering the MPLS domain at the same ingress LER, and leaving at the same egress LER, may use different routes (for instance, LSP2 and LSP3 in Figure 4.1). Therefore, an IGP adapted for TE (e.g., OSPF-TE [85] or ISIS-TE [95]) computes routes satisfying the TE constraints, while RSVP-TE is in charge of reserving resources and allocating labels along the paths. Several LSPs can be built between the same pair of LERs. Their label sequences are completely different while their IP path may or may not be distinct (for instance, LSP1 and LSP2 in Figure 4.1). An operator may also define source-routed MPLS tunnels to “manually” tune his network. In such a case, the label sequences are also likely to be specific to each LSP.

Note that RSVP-TE may be used conjointly with LDP, or not. These two protocols are independent, even if there is no reason to use only RSVP-TE for specific purposes without using LDP globally within the network.

4.3 Label Pattern Recognition Algorithm

The *Label Pattern Recognition* (LPR) algorithm classifies MPLS tunnels according to their use and the path diversity they bring. LPR does not require any additional probing to standard `traceroute`. It must be applied once the data has been collected. It classifies each <*ingress LER, egress LER*> pair into one of several classes according to the recognition of the standard

behaviors of RSVP-TE versus LDP in terms of label distribution. Those behaviors have been experimentally tested and validated in a lab environment (using both routing emulators and real routers) with different configurations and equipment.

The whole classification process is illustrated in Figure 4.2. The first step consists in extracting explicit MPLS tunnels from the `traceroute` data[5]. Then, LPR filters the different tunnels in order to remove noise, and ensure it will only focus on the diversity of transit tunnels (see Section 4.3.1). Once the data has been sanitized, the classification itself is performed. At the end, each considered *In-Out Transit Pair (IOTP)* is assigned to one of the defined classes. An *IOTP* is defined as an <*ingress LER, egress LER*> pair, i.e., a set of explicit MPLS tunnels having the same IP entry and exit points. By definition, an IOTP may have several branches, each one corresponding to a particular LSP (as illustrated on Figure 4.1, where IOTP <R_1, R_2> has three branches).

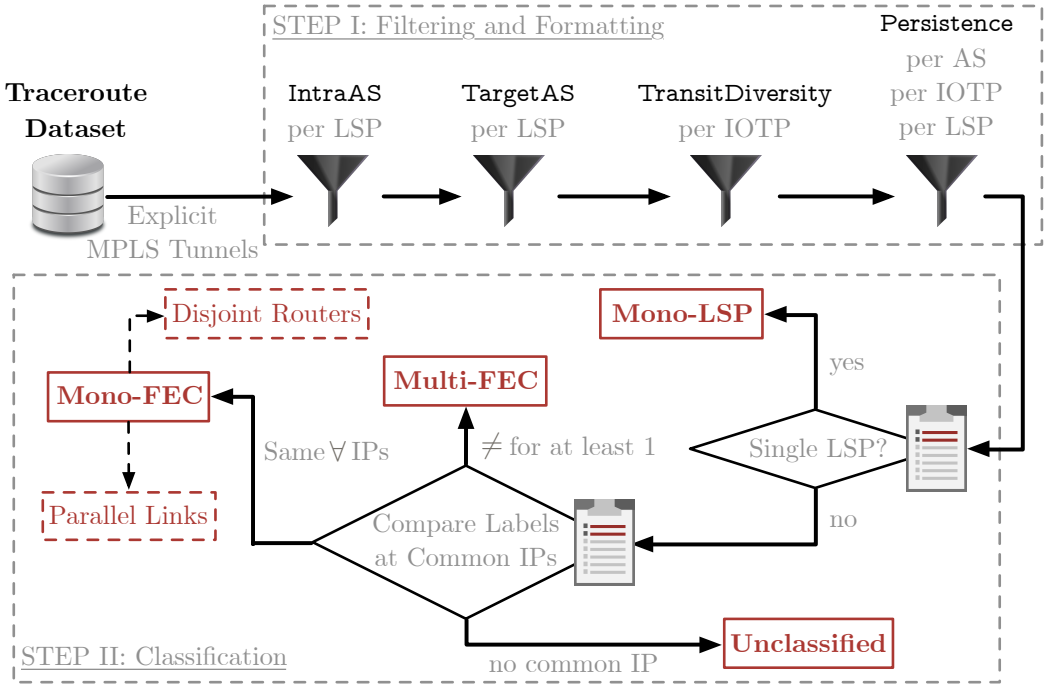


Figure 4.2: Overview of the LPR algorithm. LPR is divided into two important steps: (i) data filtering and (ii) MPLS tunnel classification.

4.3.1 Filters

The first step consists in filtering and sanitizing the set of explicit LSPs and/or IOTPs. This step is done through four different filters applied sequentially.

[5]Any `traceroute` dataset can serve as input to LPR. The only condition is to be able to retrieve explicit MPLS tunnels from `traceroute` outputs.

First, based on observations on CAIDA's dataset[6], the use of inter-domain MPLS tunnels for transit traffic is negligible. The rise in remote peering [27] does not contradict this measurement statement for two main reasons. First, remote peerings are made of invisible MPLS tunnels (i.e., no RFC4950 and `no-ttl-propagate` – see Section 2.3.4), which cannot be retrieved from the dataset. Second, remote peering consists in MPLS tunnels that belong to a remote peering provider, not tunnels going through several subsequent ASs. Therefore, inter-domain tunnels are not considered by LPR. As a consequence, all the IP addresses involved in a given LSP must belong to the same AS. In the opposite case, the tunnel is rejected. This first filtering step is done by the `IntraAS` filter (see Figures 4.2 and 4.3).

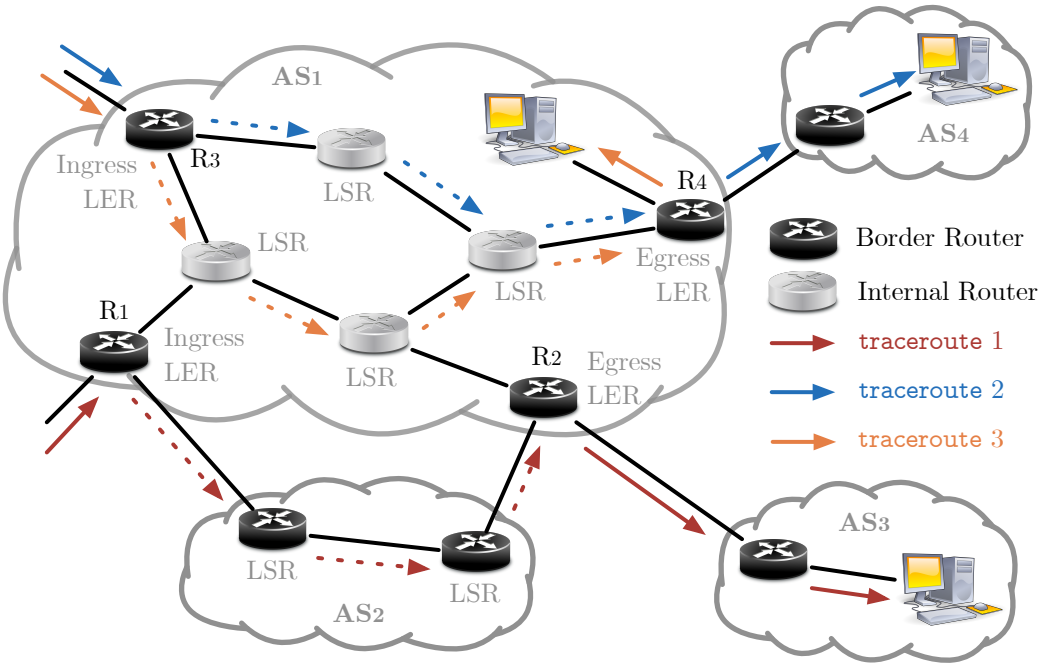


Figure 4.3: Illustration of LPR's `IntrasAS` and `TargetAS` filters. The LSP discovered by the first `traceroute` (in red) between R_1 and R_2 (an IOTP) is deleted by the `intraAS` filter, as it crosses another AS, and is then not used for transit traffic. In addition, the LSP between R_3 and R_4 revealed with the third `traceroute` (in orange) is used to reach an internal destination, and is then removed from the data by the `TargetAS` filter.

As illustrated in Figure 4.3, the objective of the `TargetAS` filter is to ensure that the `traceroute` destination is in a different AS than the tunnel itself. Indeed, in the case where they belong to the same domain, the LSP is used for an internal destination, and so, not for transit purpose.

To have a better view of the routing diversity, only the IOTPs used to reach at least two destinations belonging to different ASs are selected (`TransitDiversity` filter), as shown in

[6]The dataset will be described in Section 4.4.

Figure 4.4. The idea here is to capture multiple-FEC scenarios based on IP destination prefixes that, by definition of IP routing, represent the more classical practice of TE. It is worth noting that, even with that filter, the transit tunnel diversity may be underestimated (and so, the use of TE). Indeed, a network operator may decide to associate multiple destination prefixes to the same FEC, or create a class with other types of information, such as the source prefix, for example.

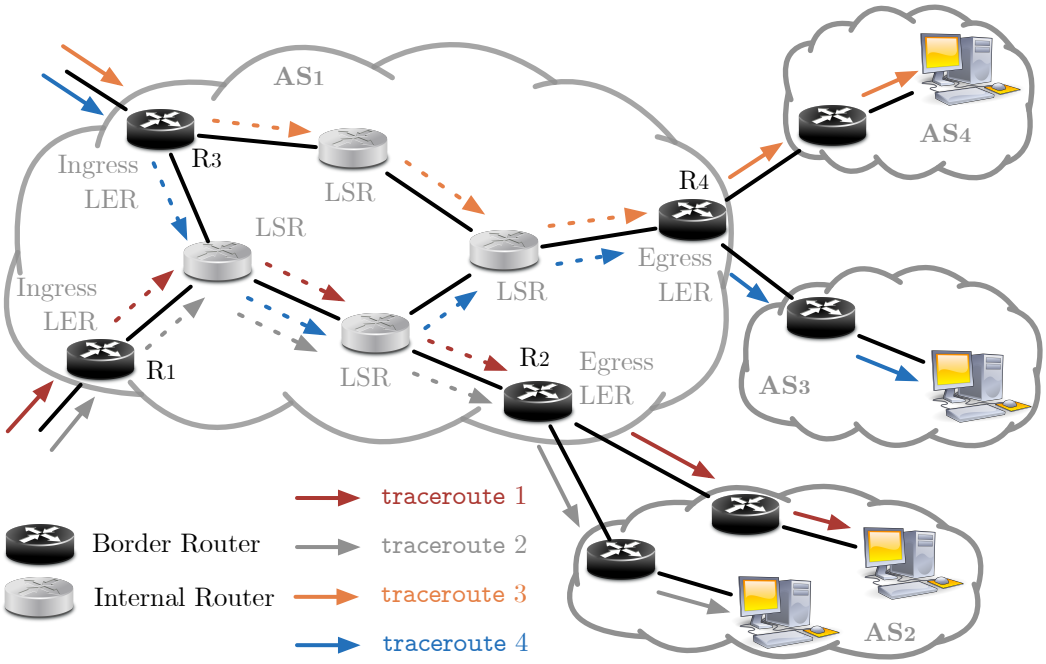


Figure 4.4: Illustration of LPR's `TransitDiversity` filter. The two first `traceroutes` (in red and grey) reveal the same LSP between R_1 and R_2. However, as their destinations are located in the same AS, the IOTP $<R_1, R_2>$ is deleted by the `TransitDiversity` filter.

Finally, the persistence over time of the LSPs is checked in order to remove the noise due to routing changes (`Persistence` filter). LPR keeps LSPs encountered in measurement cycle X only if they are also seen in measurement cycle $X+1$, $X+2$, ..., or $X+j$ (the impact of the number of additional cycles j is evaluated in Section 4.4.2). In this case, the j cycles are consecutive and taken in the same month as cycle X. Note that dynamics is also taken into account, as it can represent in itself a TE use rather than routing changes. In practice, if the vast majority of LSPs disappear for a given AS, they are reinjected in the dataset in order to perform a standard classification on the snapshot[7]. In other words, the dynamic AS is not removed from the dataset, and LPR continues to process it as the others, but adding a dynamic tag to it.

After the application of the four filters, the resulting set of IOTPs can be classified. The subsets of LSPs are robust, i.e., they are persistent in time, possibly diverse in targets, and focus on transit traffic through a single ISP.

[7]The reinjection is performed only if the whole set of LSPs was deleted by the filter.

```
# This function requires 𝕋, the set of IOTPs after filtering
def lpr_classification(𝕋)
  # Create a set for each class
  monoLSP = ∅
  multiFEC = ∅
  monoFEC = ∅
  unclassified = ∅

  for iotp in 𝕋:
    # Get all the LSPs for an IOTP
    LSPs = iotp.getLSPs()
    if (|LSPs| == 1):
      # Class 1: Mono-LSP
      monoLSP.add(iotp)
      continue
    # Search for common IP addresses
    commonIPs = iotp.getCommonIPs()
    if (|commonIPs| == 0)
      # Class 4: Unclassified
      unclassified.add(iotp)
      continue

    iotp_isMultiFEC = False
    for ip in commonIPs:
      if (|ip.getLabels()| > 1):
        # Class 2: Multi-FEC
        multiFEC.add(iotp)
        iotp_isMultiFEC = True
        continue

    if (not iotp_isMultiFEC):
      # Class 3: Mono-FEC ECMP
      monoFEC.add(iotp)

  # Return all the classes
  return monoLSP, multiFEC, monoFEC, unclassified
```

Listing 4.1: Pseudo-code for LPR classification step.

4.3.2 Classification

The pseudo-code for the classification is given in Listing 4.1. In the following, in order to determine the actual use of MPLS tunnels, the content of all LSPs belonging to the same AS, and having the same couple of border edge routers <*ingress LER, egress LER*>, is compared both in terms of IP addresses and labels.

The first class, illustrated in Figure 4.5, is called *Mono-LSP*, and gathers IOTPs having only a single LSP (i.e., same IP addresses and same labels). For these IOTPs, no transit tunnel diversity is revealed, as the same LSP is always used, whatever the destination. It means that LPR does not observe any ECMP load balancing, nor the deployment of several FECs allowing to reach different ASs with specific routing constraints, although at least two destination ASs are considered (ensured at the filtering step). The condition to reveal such a class is described at line 12 of Listing 4.1.

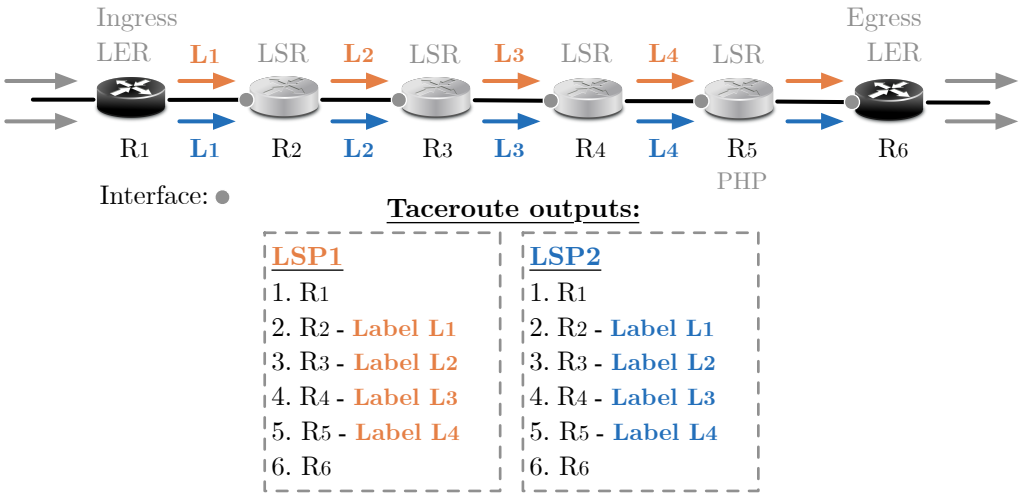


Figure 4.5: Label pattern for the Mono-LSP class. Labels and IP addresses are similar at each internal MPLS hop in both traces.

The second class, *Multi-FEC*, is illustrated in Figure 4.6. IOTPs belonging to this class have at least two LSPs converging at a given LSR. These LSPs have then a common IP interface. However, they use different labels at this specific point. For instance, in Figure 4.6, the first LSP (orange line) considers label L_1, while the second (blue line) considers label L_2. Labels L_1 and L_2 are observed on the same IP address, and thus on the same router, which suggests the use of multiple FECs. Indeed, when LDP is used as the signaling protocol, by default, labels have a router scope, as each LSR announces the same value for a given destination to all its upstream routers. As explained in Section 4.2.1, in practice, such destinations are generally loopback addresses of BGP edge routers, for scalability reasons. Therefore, distinct labels advertised by the same LSR for a given egress LER indicate distinct FECs. This use of multiple FECs suggests a TE practice for that particular IOTP. In order to provide an upper bound of the TE use, an IOTP is classified as multi-FEC as soon as distinct labels appear on a given common IP address, as described by lines 24 to 29 in Listing 4.1.

Note that the concept of common IP address is fundamental for the classification. Nothing can be concluded if none exists for a given IOTP. Fortunately, most LSPs of an IOTP converge at some point. In the LPR algorithm, a set with the common IP addresses is computed independently for

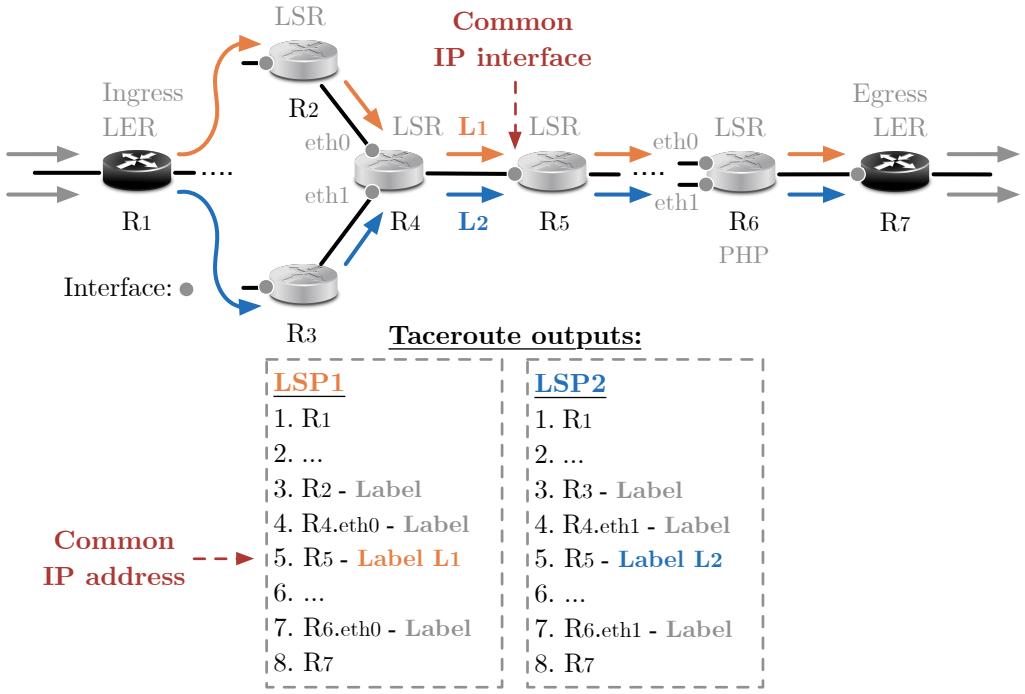
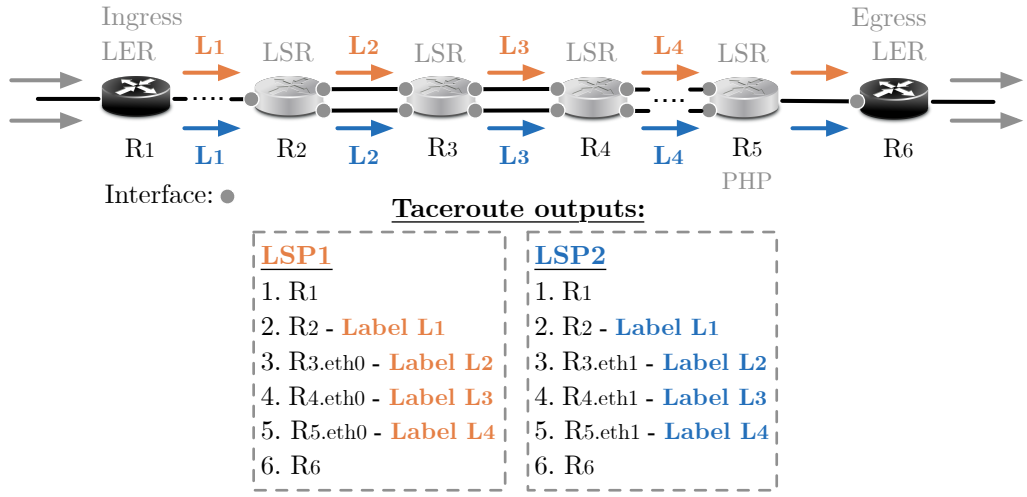


Figure 4.6: Label pattern for the Multi-FEC class. Labels are different for at least one IP address which is common to both LSPs. Labels at other LSRs are not relevant.

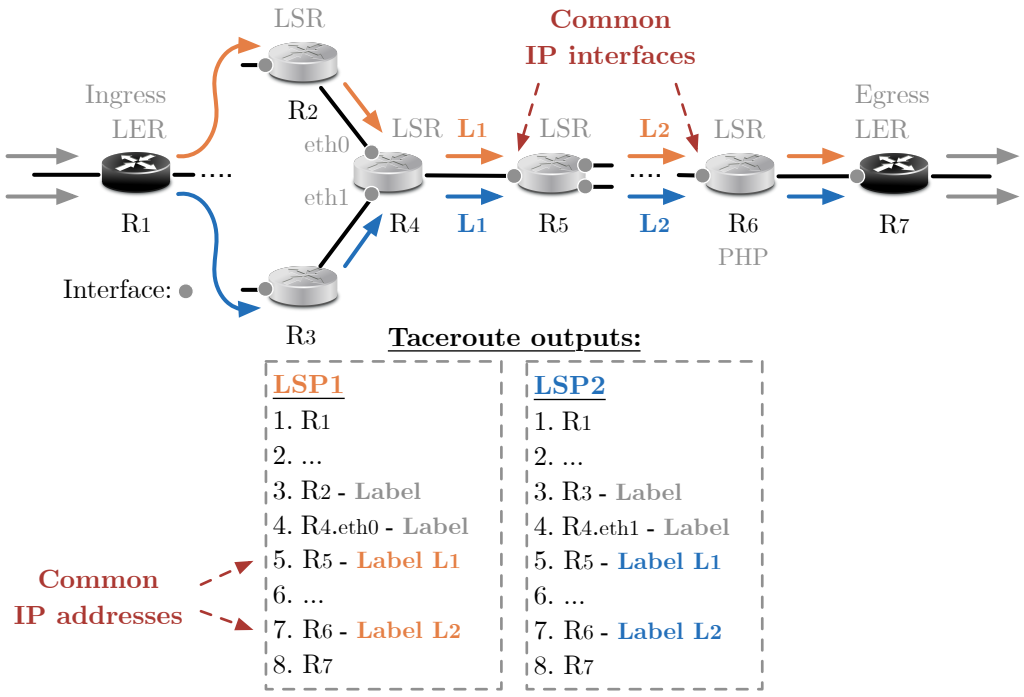
each IOTP (line 17 of Listing 4.1). It simply consists of all IP addresses belonging to LSRs that are crossed by at least two distinct LSPs of the IOTP. The two LSPs may differ at a given hop either in terms of IP addresses (this hop is then not included in the common IP set by definition) or in terms of labels (at any hop, including a common IP one).

The third class, shown in Figure 4.7, refers to IOTPs that perform load balancing between their LSPs. More specifically, all LSPs of a given IOTP crossing a common LSR use an identical label at this point, in opposition to the Multi-FEC class, where at least one difference exists. Therefore, a single FEC is used between the ingress and egress LERs, which corresponds typically to the use of MPLS load balancing on top of IP thanks to ECMP. Note that an IOTP falls in this category if and only if all its common IP addresses verify this behavior, as stated by lines 31 to 33 in Listing 4.1. This class, called *ECMP Mono-FEC* (or *Mono-FEC* for readability reasons, as it consists in Multi-LSP Mono-FEC), may be divided into two subclasses. First, if all IP addresses differ along the LSPs, but labels are identical, the addresses at a given hop are probably aliases[8]. Indeed, as the scope of LDP labels is local to the LSR, it is unlikely that two distinct devices announce the same label. It is then legitimate to conclude that the IOTP uses parallel links to perform ECMP load balancing (Figure 4.7(a)). On the contrary, if LSPs differ both in labels and IP addresses, on at least a given hop, the IOTP seems to perform ECMP load balancing through

[8]Aliases are IP addresses that belong to the same router.



(a) Parallel links.



(b) Disjoint routers.

Figure 4.7: Label patterns for the Mono-FEC class. In case of parallel links (a), IP addresses may be different, but labels are similar for each internal MPLS hop in both traces. For disjoint routers (b), labels are equal for all the IP addresses which are common to both LSPs. Labels at other LSRs are not relevant.

disjoint routers (Figure 4.7(b)). This distinction is of the highest interest: no active resolution probing is necessary to show that a large portion of ECMP load balanced paths relies actually only on parallel links.

Finally, the set of common IP addresses may be empty for a given IOTP if the LSPs converge only at the egress LER. In this situation, the IOTP is arbitrarily tagged as *Unclassified*, as indicated by lines 18 to 20 in Listing 4.1. In practice, this situation rarely occurs, as it will be demonstrated in Section 4.4.

4.4 Evaluation

In this section, we will study the application of LPR on data collected by CAIDA in order to evaluate and understand the use of MPLS on the Internet. After a description of the measurement data, the impact of the different filters will be estimated. We will also analyze several IOTPs properties, and focus on a subset of five large ASs (mainly Tier-1) whose MPLS deployment is characteristic. Finally, we will discuss some properties of MPLS label dynamics on a specific AS.

4.4.1 Dataset

In order to evaluate the use of MPLS and how it has evolved over time, LPR was applied on an extensive dataset obtained from CAIDA's Archipelago infrastructure. In this platform, three teams of monitors scattered around the world run regularly `paris-traceroute` [10] measurements (with ICMP probes) to all IPv4 routed /24 prefixes in a short amount of time. Currently, the infrastructure is made of more than 200 monitors.

The data considered in this analysis was collected between January 2010 and December 2014. Only the first run of each team (called a *cycle*, or *snapshot*) was kept for each month in that period. For all the 60 cycles, an IP-to-AS mapping was performed using Routeviews data [142] collected the corresponding days. Then, all explicit tunnels were extracted from each snapshot.

Figure 4.8 shows, for each cycle, the proportion of traces that cross at least one explicit tunnel (before any kind of filtering performed by LPR's cleaning step). During the five years of data, the deployment of MPLS increased significantly. Around the 30^{th} cycle, a rise of roughly 10% of the number of traces showing at least one MPLS tunnel can be observed. On the contrary, their proportion tend to decrease during the last months. This phenomenon is due to modifications of the MPLS use inside AS3356 (see Section 4.4.4, and Figure 4.21 in particular, for details about AS3356, a Tier-1 network named LEVEL3). Figure 4.9 shows, for each cycle, the raw number of unique IP addresses used in MPLS tunnels (upper graph), and those that aren't (lower graph). Even if only a slight increase of the total number of addresses can be observed in the collected data (21% over the five years), the number of those appearing in MPLS networks grows quite fast (60%). Note that the two drops at cycles 23 and 58 are caused by measurements issues in the Archipelago infrastructure.

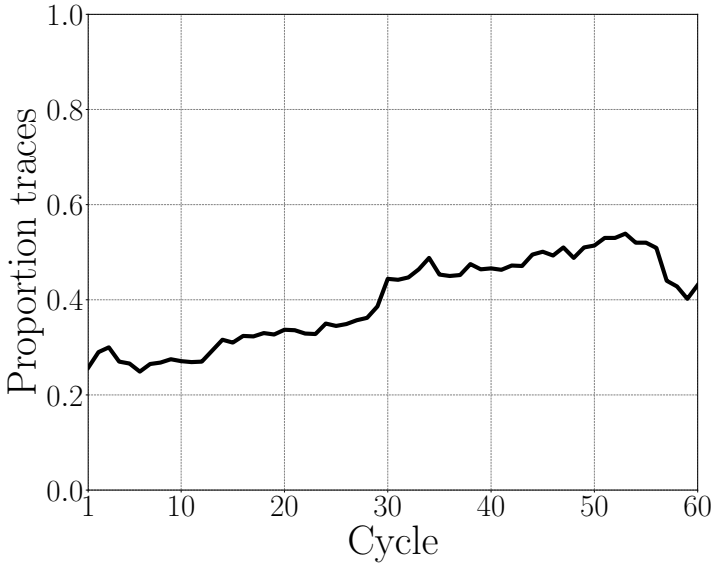
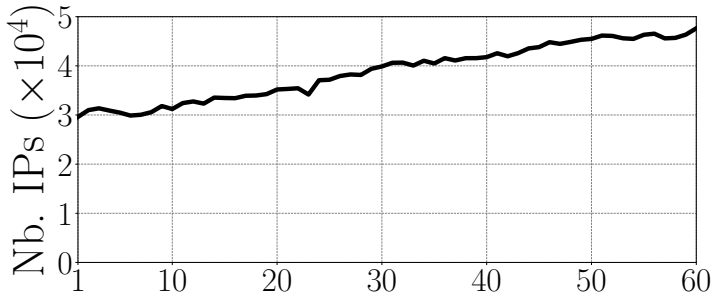
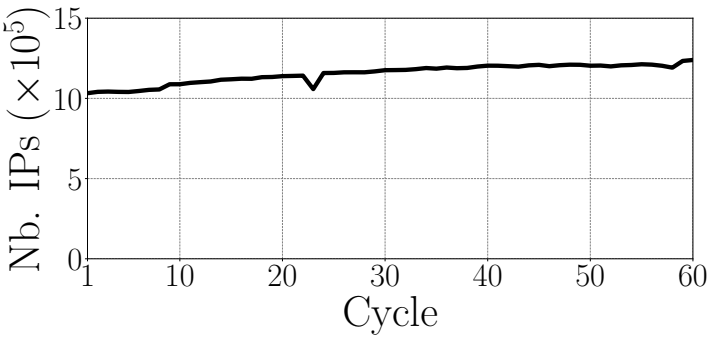


Figure 4.8: Proportion of traces crossing at least one explicit MPLS tunnel.



(a) Addresses flagged as MPLS.



(b) Addresses never flagged as MPLS.

Figure 4.9: IP addresses observed in the dataset.

Globally speaking, and as already stated previously [53, 137], this evolution shows that MPLS is being deployed more and more during the last years. This increase confirms the interest of many operators in MPLS, and opens the question of its actual use.

4.4.2 Filtering Impact

In this section, we will evaluate the impact of LPR's filtering process (see Section 4.3.1). Table 4.1 gives a first overview of the filtering effects on LSPs. It provides, over the 60 cycles, the average proportion (with confidence interval) of LSPs that remains after each filter. The line "Incomplete LSPs" refers to LSPs that are incomplete in the `traceroute` sense, i.e., at least one of their LSRs did not reply to probes (i.e., anonymous routers). These LSPs are removed from the dataset, as they can not be classified by the algorithm due to the missing information. The filter deletes on average 14.7% of LSPs, and has then the strongest impact on the dataset. It is not surprising, as incomplete traces due to anonymous routers are not rare [75]. Moreover, this filter being the first one to be applied, its effect is more important compared to others.

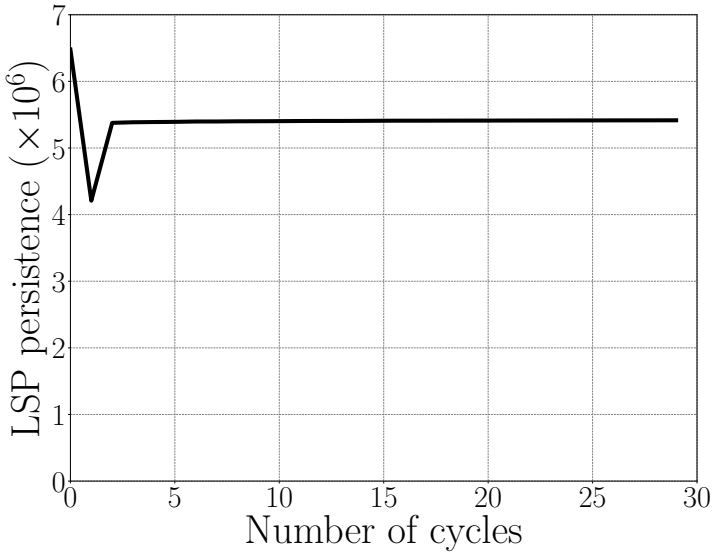
Filter	**Average**
Incomplete LSPs	0.853 ±0.01
`IntraAS`	0.844 ±0.01
`TargetAS`	0.717 ±0.009
`TransitDiversity`	0.644 ±0.009
`Persistence`	0.534 ±0.007

Table 4.1: Impact of LPR's filtering step on the dataset. The table presents, over the 60 cycles, the average proportion (and confidence interval) of LSPs that remains after each filter. On average, a cycle contains 14×10^6 LSPs before filtering.

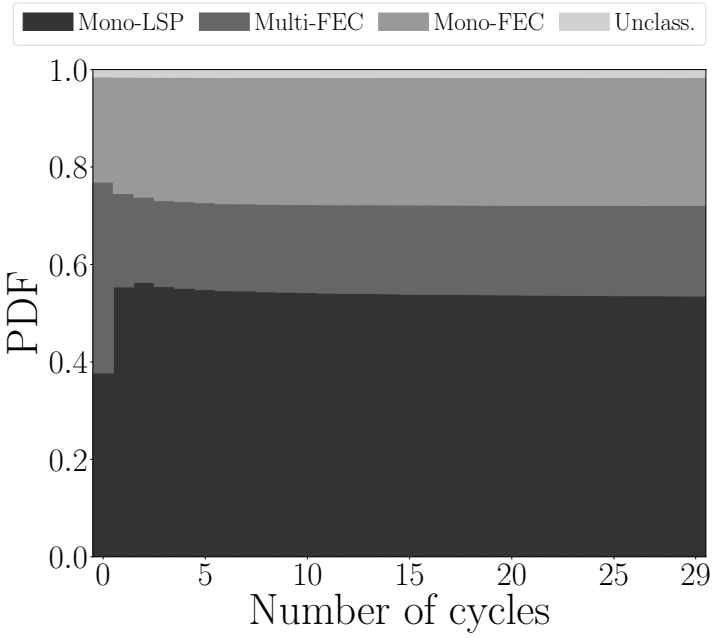
Section 4.3.1 announced that the number of inter-domain MPLS tunnels is negligible. This statement is confirmed by the statistics associated to the `IntraAS` filter, as only a low 0.9% of the LSPs concerns inter-domain tunnels. Further, on average, 12.7% of the LSPs are used to forward packets to an internal destination located in the same AS as the tunnel itself (`TargetAS`). Then 7.3% of the tunnels allow to reach only a single destination (`TransitDiversity`). Finally, 10% of the remaining LSPs are removed due to routing noise (`Persistence`). This last filter keeps LSPs encountered in measurement cycle X only if they are also seen at least once in measurement cycle $X+1$, $X+2$, ..., or $X+j$. The j snapshots are taken consecutively in the same month as cycle X. To compute the values in Table 4.1, j had a value of 2.

The effect of the `Persistence` filter depends on the number of subsequent cycles (j) in which each LSP must be identified at least once. The impact of the value of j was analyzed based on on the 29 Archipelago snapshots composing the whole December 2014. During the analysis, the parameter j varied between 0 (no filtering) and 29 (the whole month is used for the persistence

analysis[9]). Results are shown in Figure 4.10.



(a) Number of LSPs at the filter output, depending on the number of considered subsequent cycles.



(b) Classification according to the number of considered subsequent cycles.

Figure 4.10: Impact of the `Persistence` filter on the December 2014 dataset cycle.

Figure 4.10(a) depicts the amount of remaining LSPs after the application of the `Persistence` filter. Increasing the parameter j does not affect that much the number of LSPs. When $j \geq 2$, their

[9]An Archipelago cycle may last a bit more than one day, leading to a total of 29 snapshots for the month, instead of 31.

amount remains mostly stable. The highest value is observed when $j = 0$, which is expected as, by definition, no `Persistence` filter was applied. In the same fashion, the large drop at $j = 1$ is due to the necessity to retrieve the LSP in the next snapshot, while the condition is more relaxed for $j \geq 2$ (the LSP must appear in one of the considered subsequent snapshots). Figure 4.10(b) shows the impact of the `Persistence` filter on the IOTP classification, at a global scale. When $j \geq 2$, the classification remains stable. The effect of selecting $j \leq 1$ is to trade Mono-LSP tunnels with Multi-FEC tunnels, which is partially explained by the use of dynamic Multi-FEC LSPs (see Section 4.4.5).

In the remainder of this chapter, the parameter j of the `Persistence` filter is set to 2, meaning that an LSP observed in cycle X will be considered for classification if it is also encountered in measurement snapshot $X + 1$ or $X + 2$ of the same month.

4.4.3 Tunnel Length, Width, and Symmetry

In order to describe the observed IOTPs at a high level, the metrics suggested by Augustin et al. [11] for load balanced paths will be adapted to MPLS. This adaptation is illustrated in Figure 4.11. As a reminder, an IOTP is an <*ingress LER, egress LER*> pair corresponding to a set of explicit MPLS tunnels having the same entry and exit points.

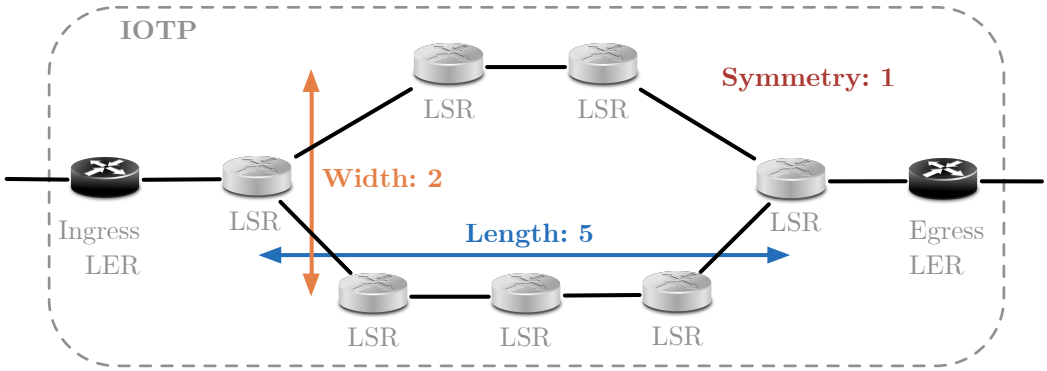


Figure 4.11: Illustration of the IOTP metrics.

The *length* of an IOTP is defined as the number of LSRs in the longest LSP of that IOTP, without counting the ingress and egress LERs. Therefore, it corresponds to the number of intermediate LSRs in the longest LSP. Figure 4.12 provides the length distribution of the IOTPs observed in the last cycle of the dataset (i.e., December 2014). In most cases (i.e., more than 65%), paths are rather short, with less than 4 LSRs[10], although a very low proportion of them can be quite long. This property is related to the short diameter of many ASs [100].

Figure 4.13 shows the IOTP *width* distribution, meaning the number of "branches" between the ingress LER and the egress LER. Branches may differ physically (i.e., IP addresses) or

[10]To which the LERs must be added to obtain the complete tunnel.

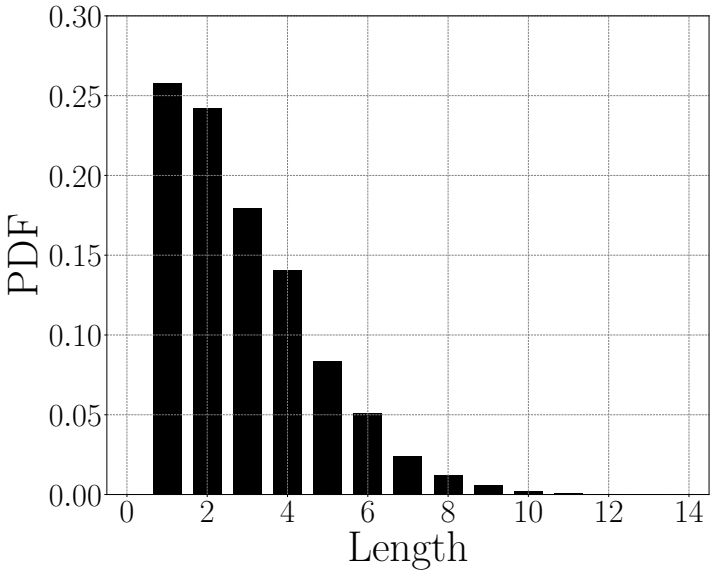


Figure 4.12: IOTP length distribution (December 2014).

logically (i.e., MPLS labels). By definition, only tunnels falling in the Mono-LSP class will have a width of 1. Although a very low proportion of the IOTPs are very large (up to 236 branches in the dataset), most of them (56%) are narrow, with a width of 1. In the dataset, the observed IOTPs are thus mainly Mono-LSP. This result is consistent with Figure 4.10(b), in which the majority of IOTPs are indeed in the Mono-LSP class. Additionally, Figure 4.10(b) also reveals how tunnels are used in general. The deployment of RSVP-TE (and thus TE) is limited (roughly 20% of the IOTPs), leading thus to the conclusion that tunnels are essentially built based on LDP.

Figure 4.14 shows the width distribution for each class separately, except the Mono-LSP one[11]. The key point here is that Mono-FEC and Multi-FEC tunnels have nearly the same distribution. Although the tail of the plot is slightly dominated by Multi-FEC, this result is quite surprising as it tends to indicate that TE-based LSPs do not use much more path diversity than basic ECMP (with a different way to perform load balancing though). Since the width distribution is biased by the measure itself (it represents a lower bound as the measurement campaign did not provide a complete exploration), the following property may be useful to better understand structural differences between these two kinds of path diversity.

The *symmetry* of an IOTP is defined as the difference between its length and the number of LSRs in its smallest LSP. If the symmetry equals 0, the IOTP is said *balanced* (or *symmetrical*). Otherwise, it is considered as *unbalanced* (or *asymmetrical*), and the obtained value gives an idea of the imbalance. Obviously, by definition, an IOTP in the Mono-LSP class is balanced. Therefore, the distribution given in Figure 4.15 does not show this class[12]. The first observation

[11]Unclassified tunnels (not shown on Figure 4.14 as they are marginal – see Figure 4.10(b)) are mainly narrow.
[12]Unclassified IOTPs are marginal, and thus, not shown in Figure 4.15.

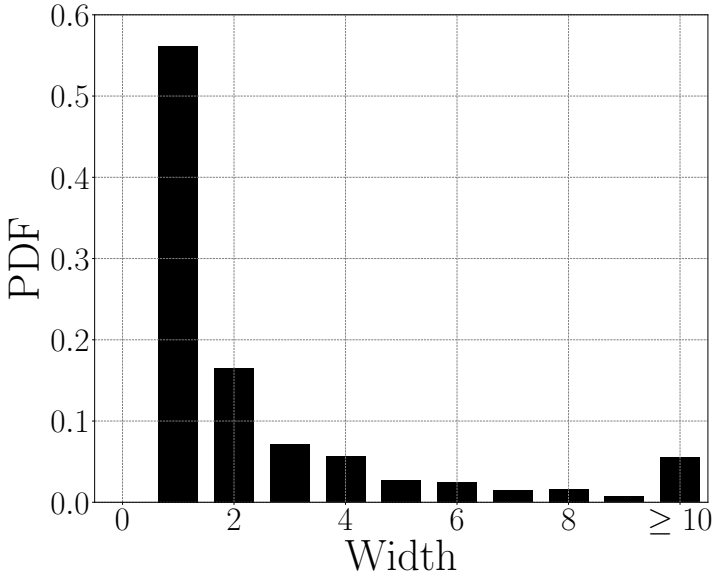


Figure 4.13: IOTP width distribution (December 2014).

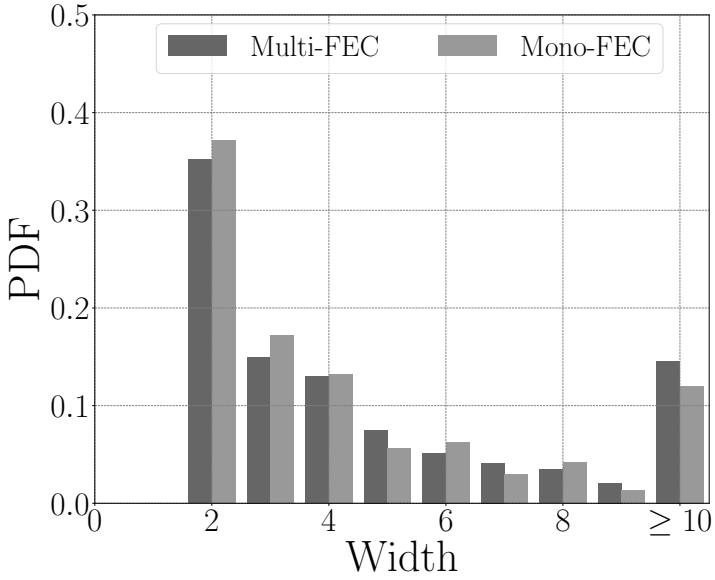


Figure 4.14: IOTP width distribution for the Mono-FEC and Multi-FEC classes (December 2014).

is that balanced paths represent 80% of the distribution for both classes. This property is derived from two facts. First, ECMP forwarding paths are often similar in terms of hop count (80% of Mono-FEC IOTPs are balanced), and many multi-FEC LSPs actually go through the same path, and differ only in terms of labels (this observation is the main cause of the 80% of balanced Multi-FEC IOTPs). Then, again, there are no significant differences between the Mono-FEC and the Multi-FEC classes. This similarity is really surprising, as one may speculate that TE paths may completely differ among themselves according to each FEC constraint. In practice, constraints seem to be satisfied by a unique IP path in most cases, meaning that reserving bandwidth in an over-provisioned network is equivalent to using resource pooling mechanisms such as ECMP.

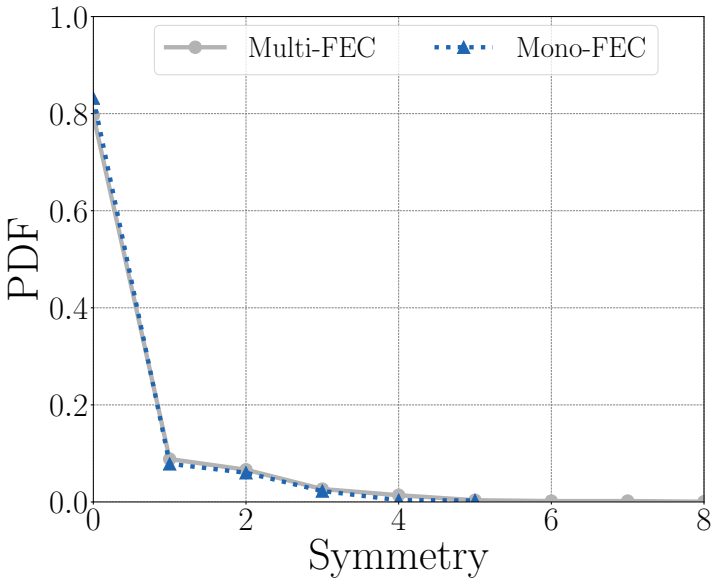


Figure 4.15: IOTP symmetry distribution for the Mono-FEC and Multi-FEC classes (December 2014).

4.4.4 IOTPs Classification

Figures 4.16, 4.17, 4.18, 4.19, and 4.20 present the classification of the IOTPs of fives ASs that are representative of typical MPLS behaviors across the whole dataset. It is worth noting that they are all Tier-1 ASs, except VODAFONE (AS2173), which is a transit network. Each of those figures is made of two parts:

- The lower part gives, for each cycle (X-axis), the number of IOTPs considered for the classification.
- The upper part provides, for each cycle (X-axis), the proportion of tunnels in each of the four classes.

		AS1273 (VODAFONE)		**AS7018** (AT&T)		**AS6453** (TATA)		**AS2914** (NTT)		**AS3356** (LEVEL3)	
		~~MPLS~~	MPLS	~~MPLS~~	MPLS	~~MPLS~~	MPLS	~~MPLS~~	MPLS	~~MPLS~~	MPLS
2010	min	796	0	36,614	379	2,099	435	2,760	191	9,121	0
	max	1,097	160	37,656	588	2,269	513	3,061	247	9,417	14
	avg	887	115	36,984	493	2,181	463	2,884	216	9,253	1
2011	min	828	108	34,537	555	1,896	326	3,128	226	9,245	0
	max	914	192	39,988	775	2,209	454	3,368	270	9,775	0
	avg	871	147	37,874	663	2,043	380	3,243	248	9,503	0
2012	min	901	149	40,022	608	1,883	291	3,293	243	9,464	0
	max	993	230	48,839	701	2,118	334	3,429	293	10,033	646
	avg	950	199	45,764	655	2,005	308	3,349	262	9,700	369
2013	min	918	134	48,946	554	2,017	260	3,349	283	9,434	570
	max	1,014	234	57,995	648	2,165	336	3,641	314	9,997	726
	avg	971	178	50,597	605	2,084	298	3,490	300	9,708	676
2014	min	928	147	50,518	486	2,054	268	3,652	208	9,342	3
	max	1,088	222	57,403	554	2,162	314	4,116	328	11,108	776
	avg	990	171	52,489	508	2,099	293	3,885	316	10,162	518

Table 4.2: Statistics about the IP addresses of some ASs of interest (after filtering). The column ~~MPLS~~ refers to non-MPLS IP addresses.

Empty zones in both parts of a graph refer to cycles where no MPLS tunnel was encountered. Finally, note that those figures must be read conjointly with Table 4.2, which provides the name of the analyzed network infrastructures, as well as some statistics about them (i.e., minimum/-maximum/average number of IP addresses, tagged as MPLS or not, observed each year).

Figure 4.16 gives the classification for AS1273 (VODAFONE, large transit ISP). AS1273 is interesting in several points of view. First, its use of MPLS (for transit traffic) has increased over time (see below part of the figure). It is confirmed by the values in Table 4.2. Second, this AS seems to deploy MPLS mostly for TE reasons. The Multi-FEC class grows with time to the detriment of the Mono-LSP use. In addition, the Mono-FEC (ECMP) class is almost invisible.

The classification for AS7018 (AT&T, Tier-1 ISP) is presented in Figure 4.17. The use of MPLS decreases relatively over time, while the Multi-FEC class is more and more privileged in comparison with Mono-FEC tunnels. A drop in the number of IOTPs can be observed around cycle 22. It seems to correspond to a transition in the MPLS use.

Figure 4.18(a) shows the classification for AS6453 (TATA COMMUNICATIONS, Tier-1 ISP). This AS has almost no Multi-FEC tunnels. However, it heavily uses ECMP with load balancing, as a large (although declining) proportion of its IOTPs fall in the Mono-FEC class. As explained in Section 4.3.2, this class can be split into "Routers Disjoint" (i.e., LSPs in a given IOTP differ both in labels and IP addresses on at least one hop) and "Parallel Links" (i.e., the labels of all LSPs of a given IOTP are identical at each hop, while the corresponding IP addresses are different).

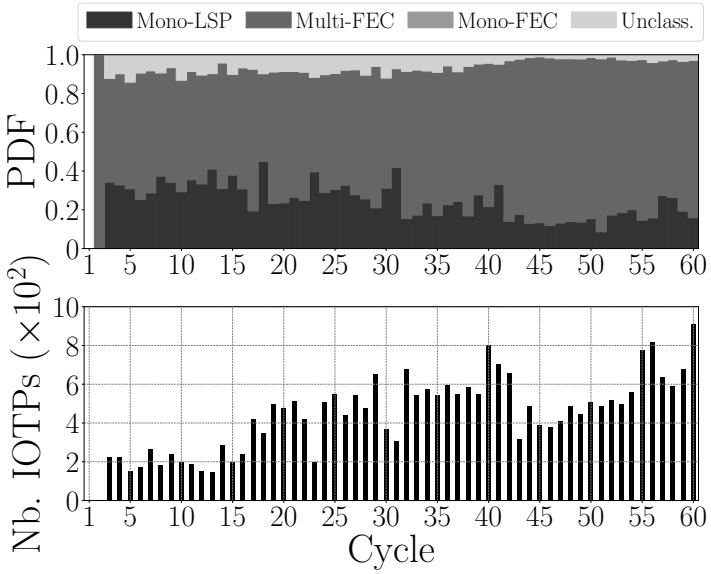


Figure 4.16: Classification for AS1273 (VODAFONE).

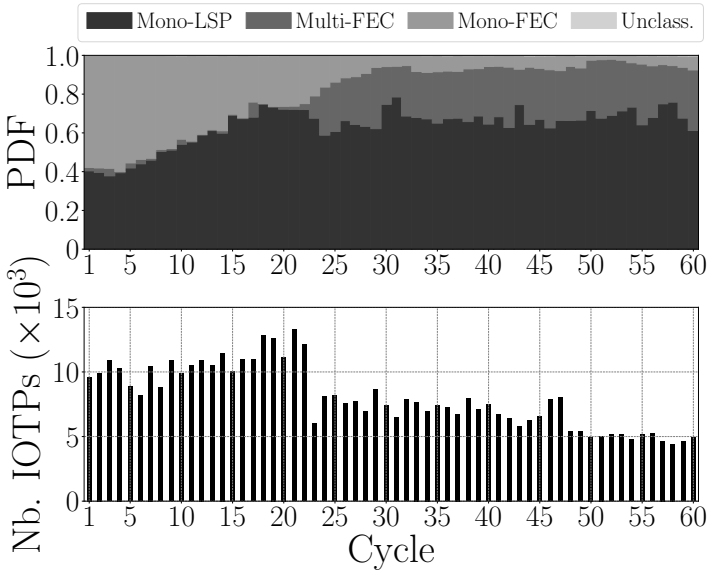
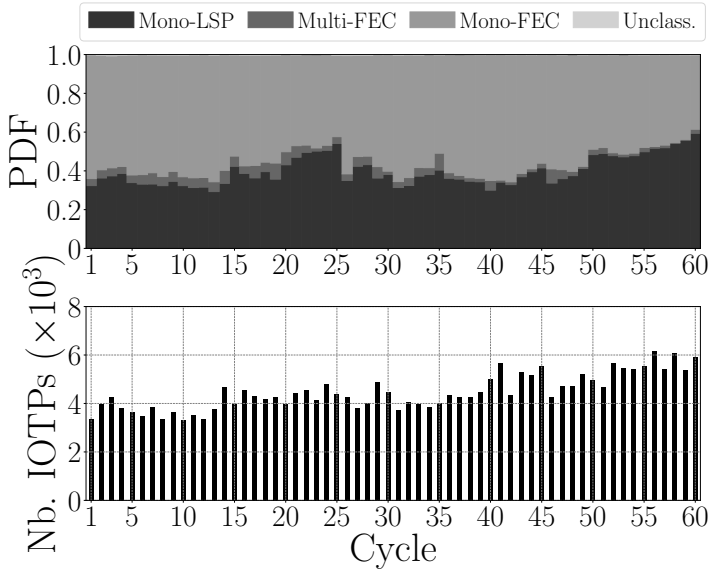
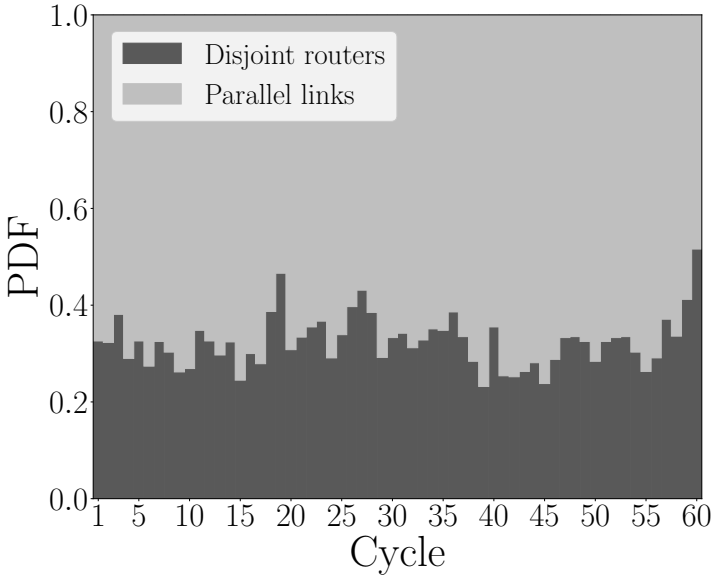


Figure 4.17: Classification for AS7018 (AT&T).

Figure. 4.18(b) shows that AS6453 has deployed Mono-FEC tunnels over time, mostly based on parallel links (between 60 and 70% of tunnels belong to the Parallel Links subclass).



(a) Classification.



(b) Distinction between “Routers Disjoint” and “Parallel Links” for the Mono-FEC class.

Figure 4.18: Classification for AS6453 (TATA COMMUNICATIONS).

The classification for AS2914 (NTT, Tier-1 ISP) is depicted in Figure 4.19. This AS shows an increase in its use of MPLS. The number of IOTPs observed during the period of interest has been multiplied by three, which is consistent with the fact that the number of IP addresses used for MPLS has also increased (see Table 4.2). The main class (Mono-LSP) seems mostly stable

over time, even if its proportion slightly decreases to the benefit of Mono-FEC.

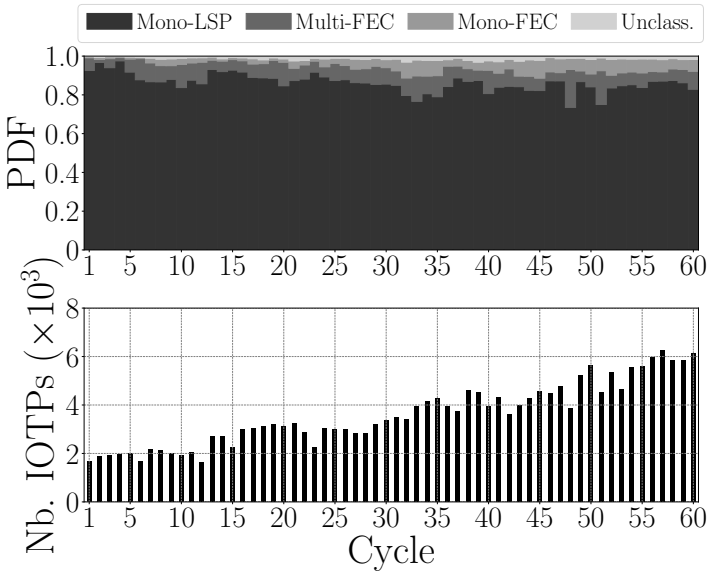


Figure 4.19: Classification for AS2914 (NTT).

Figure 4.20 gives the classification for AS3356 (LEVEL3, Tier-1 ISP), which has a quite curious shape: MPLS appears during the 29$^{\text{th}}$ cycle, observes a period of stability, and finally, presents a sharp decrease starting at cycle 55. The data before cycle 29 was investigated in order to determine whether the rise of MPLS during that cycle matches with a new MPLS hardware (or IP addresses) deployment, or a dramatic change in routing. Looking at the Archipelago files, in an infrastructural point of view, nothing has changed between cycles 28 and 29, i.e., the same set of IP addresses is observed. It means that only the use of the existing infrastructure has been modified between these cycles.

To investigate further, Figure 4.21 presents the state of MPLS deployment for the month prior to the 29$^{\text{th}}$ cycle (April 2012). To do so, all daily Archipelago data dumps were downloaded for April 2012, without respecting the probing cycle (as described in Section 4.4.1). It implies that the number of considered Archipelago vantage points differs from one day to another. The figure provides two pieces of information: (i) the number of IOTPs encountered each day, before and after the filtering steps (upper part of the figure), and (ii) the number of LSPs identified each day, again before and after the filtering steps (lower part of the figure). Peaks and drops observed in the number of IOTPs observed after April 25$^{\text{th}}$ are due to the variations in the number of considered Archipelago vantage points.

We can observe that the deployment of MPLS started around April 15$^{\text{th}}$. It took a half month to fully deploy MPLS in AS3356. The network exhibits an incremental MPLS deployment rather than an abrupt transition. Further, the number of LSPs does not differ before and after filtering (the `Persistence` filter is not used here), while the number of IOTPs does. The reason is that

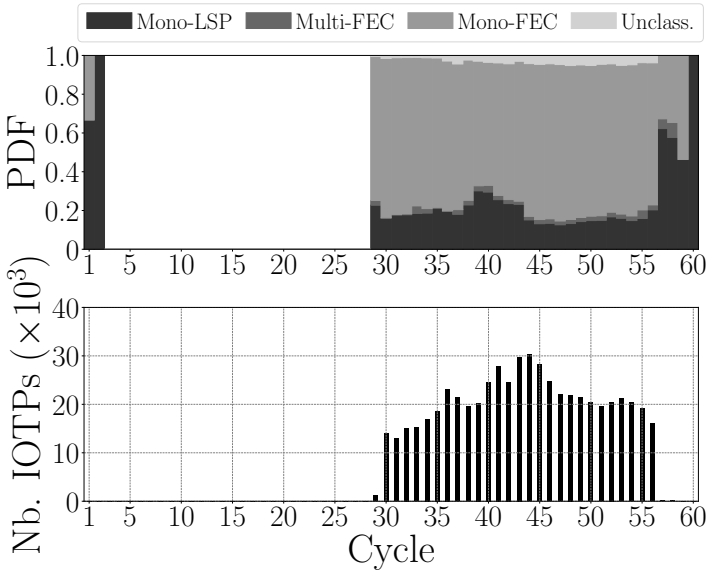


Figure 4.20: Classification for AS3356 (LEVEL3).

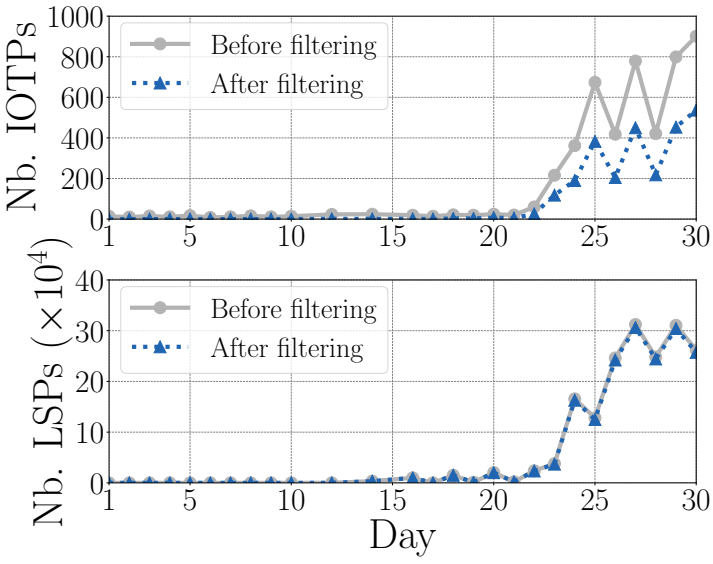


Figure 4.21: MPLS deployment in AS3356 (LEVEL3) in April 2012.

most LSPs are "shared" by several IOTPs, i.e., only one out of the two LSP extremities (the LERs) differ among a subset of them in many cases.

Generally speaking, we observed in this section that using LDP is the rule, with, or without ECMP (Mono-FEC versus Mono-LSP), while RSVP-TE seems to be more marginal both in terms of deployment, and more surprisingly, the structural path diversity it enables compared to ECMP.

4.4.5 Multi-FEC and Label Dynamics

The `Persistence` filter (see Section 4.3.1) was introduced to remove the side effects of routing dynamics. For example, a routing change during the measurements could lead to infer two LSPs in a given IOTP (hence a Multi-LSP IOTP in theory), while they are not used simultaneously in practice. However in some (rare) cases, this filter could hide an interesting phenomenon, namely LSPs that change their labels very frequently. These LSPs are not deleted by the filter, so that the corresponding ASs can be studied in a different way. In practice, they are easily retrievable since all (or almost all) LSPs would normally be filtered[13].

In order to investigate this behavior a little bit further, a small number of ASs tagged as dynamic was selected (AS1273 is one of them). An additional `paris-traceroute` campaign (with ICMP probes) was then run from a unique vantage point located in Strasbourg, the purpose being to monitor the frequency of the label changes. So, each destination was traced multiple times, at a very high sampling rate, i.e., a `traceroute` every two minutes. The results for a single LSP made of two LSRs are given in Figure 4.22. The Y-axis represents the label values (they range from 300,000 to 800,000), and each line represents their evolution for a given LSR. When a label reaches its maximum, it starts again from the minimum.

The figure shows that the LSP is reconstructed at a high frequency, and almost periodically. A possible interpretation is that it is built with RSVP-TE, and that the ingress LER is configured to "re-optimize" frequently the LSP (although in practice, it follows the same IP route). Moreover, three interesting findings can be highlighted. First, this systematic temporal-based behavior seems to be mainly related to JUNIPER hardware[14]. Second, the label changes seem to be performed on a factual basis (on each curve, some step durations differ at some points). Eventually, the shape of the curve resulting from LSR_2 evolves more quickly than the one of LSR_1. It suggests that LSR_2 is more solicited than LSR_1 in terms of number of LSPs going through it.

4.5 Conclusion

This chapter presented a new algorithm, LPR, that takes data from `traceroute` campaigns as input, and produces a classification of MPLS paths in transit networks. It allows to get some

[13] Remember that if the vast majority of LSPs disappears for a given AS due to the `Persistence` filter, they are reinjected in the dataset in order to perform the classification. The difference with the standard classification is that those particular ASs are tagged as dynamic.

[14] This statement was verified with the fingerprinting method presented in Chapter 3.

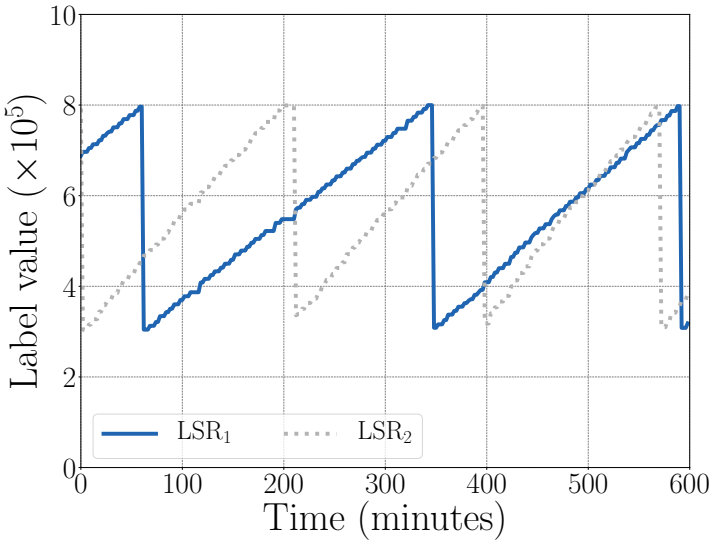


Figure 4.22: Label range evolution for the two internal LSRs of a Multi-FEC tunnel belonging to AS1273 (VODAFONE), as seen from a vantage point located in Strasbourg.

insights on the different uses of MPLS. More specifically, it is able to determine if the protocol is deployed either as a basic encapsulation technique allowing, through LDP, scalability and independence between routing/forwarding in the ISP network and outside, or, as a more refined technique, allowing, through RSVP-TE and Traffic Engineering (TE), to differentiate traffic and finely tune packet forwarding.

We analyzed a classification obtained by applying LPR over five years of data collected by the Archipelago platform. We observed that the use of MPLS is increasing over this period, and that the basic encapsulation method seems predominant, with or without path diversity. Although the deployment of RSVP-TE is less common, traffic engineering is also well represented in some ASs. Another (not so surprising) lesson, is that the use of MPLS varies greatly depending on the considered AS, from almost only mono-paths (no diversity) to a wide deployment of traffic engineering. For a given network, the class distribution may also change a lot over time.

Since most routers implement MPLS capabilities, deploying (or removing) MPLS is mainly a system configuration process. Similarly using LDP, RSVP-TE, or both is a matter of configuration. A last lesson is that when TE is deployed, in many cases, different LSPs take the same IP path between the same endpoints. Bandwidth seems then to be sufficiently abundant for allowing all LSPs to share the same physical route.

CHAPTER 5

MPLS DEPLOYMENT IN IPV6 NETWORKS

Recent researches [43, 51] have stated the fast deployment of IPv6. It has been demonstrated that it is more and more adopted by Internet service providers, but also by servers and end-hosts. In parallel, a few studies were conducted on the identification of MPLS tunnels in `traceroute` data[1]. However, they focused only on IPv4, where the protocol is strongly deployed. In this chapter, we will have a first look at how MPLS is used in IPv6 networks based on traces collected by CAIDA's Archipelago platform [145]. In particular, we will evaluate if its deployment and use differ between IPv4 and IPv6 based on the size and content of the label stacks. In a second time, we will also study the 6PE architecture [47], one of the main MPLS structures dedicated to IPv6 traffic. More specifically, we will check if the deployment of this mechanism, specifically designed to forward IPv6 packets through networks that are not fully IPv6-compliant, results mainly from non dual-stack routers, or from the absence of native IPv6-capable MPLS signaling protocols.

5.1 Introduction

During the last years, IPv6 has drawn the attention of the research community. For instance, Dhamdere et al. [51] demonstrated that it is differently deployed over the world (it is more present in Europe than in the USA), while its routing dynamics and path performance are largely identical to IPv4's. A few years ago, Czyz et al. [43] showed that IPv6 networks were becoming mature, and entering a production mode. Further, on September 24, 2015, ARIN's IPv4 free pool reached zero, effectively triggering full IPv4 depletion. The registry is now unable to provide any

[1] These studies have been discussed in Chapter 2.

IPv4 prefix, except for those requiring a small block in order to ease the transition to IPv6 [5]. The need to adopt IPv6 becomes then more and more critical.

In parallel to the IPv6 interest, MPLS has been investigated by researchers, as stated in Chapter 2. However, all those works focused on IPv4 networks only. In this chapter, we will analyze the state of MPLS deployment under IPv6. In particular, we will study how operators are using the protocol in their IPv6 domain, and whether this use differs from the one in IPv4. To achieve this goal, we will rely on an IPv6 `paris-traceroute` dataset collected by CAIDA's Archipelago measurement platform between 2009 and 2015. Based on the different explicit MPLS tunnels[2] extracted from the traces, we will show that, in parallel to an increase of the IPv6 adoption, MPLS is more and more deployed in IPv6 networks along the time. We will also demonstrate that it is essentially used for 6PE purpose, either for connecting IPv6 islands together, or for exploiting the IPv4 version of LDP to build tunnels carrying both IPv6 and IPv4 traffic on dual-stack MPLS routers. Finally, we will investigate the particular case of COGENT, a large Tier-1 ISP having both a very prominent position in the dataset, and a very particular behavior in regards to 6PE.

5.2 MPLS in IPv6

MPLS can be used in IPv6-only networks in the same way as in IPv4 domains[3]. Most routing and label distribution protocols [9, 46] have now their IPv6 version. However it was still not the case a few years ago. Moreover, providers do not always activate IPv6 capabilities, even when they are available in their hardware and software. Therefore, specific mechanisms have been devised to deliver IPv6 traffic across networks not supporting IPv6 routing (IPv4-only networks) or where some technologies are not IPv6-aware, such as LDP [7, 9], for example.

As introduced in Chapter 1, one of the MPLS uses is to connect IPv6 islands through an IPv4 core network that is unaware of IPv6. This mechanism, called *6PE* [47], is illustrated in Figure 5.1. In this architecture, *Customer Edge (CE)* routers belonging to IPv6 networks are connected to dual-stack *Provider Edge (PE)* routers located at the edge of the IPv4 domain. Each CE device sends its IPv6 prefixes to the PE router to which it is attached. IPv6 reachability is then exchanged between PE equipments via the *MultiProtocol - Border Gateway Protocol (MP-BGP)* [18]. In that context, the PE devices that perform 6PE are the ingress and egress LERs. When exchanging data, core routers are not aware that they are forwarding IPv6 packets, as they only use the label value to determine the next hop on the path.

Similarly to the VPN architecture (see Chapter 1), 6PE drives the need for two labels in the data plane (due to the potential use of PHP in particular):

[2] Explicit MPLS tunnels were presented in Section 2.3.1.

[3] See George et al. [67] for discussion on the gaps remaining between IPv4 and IPv6.

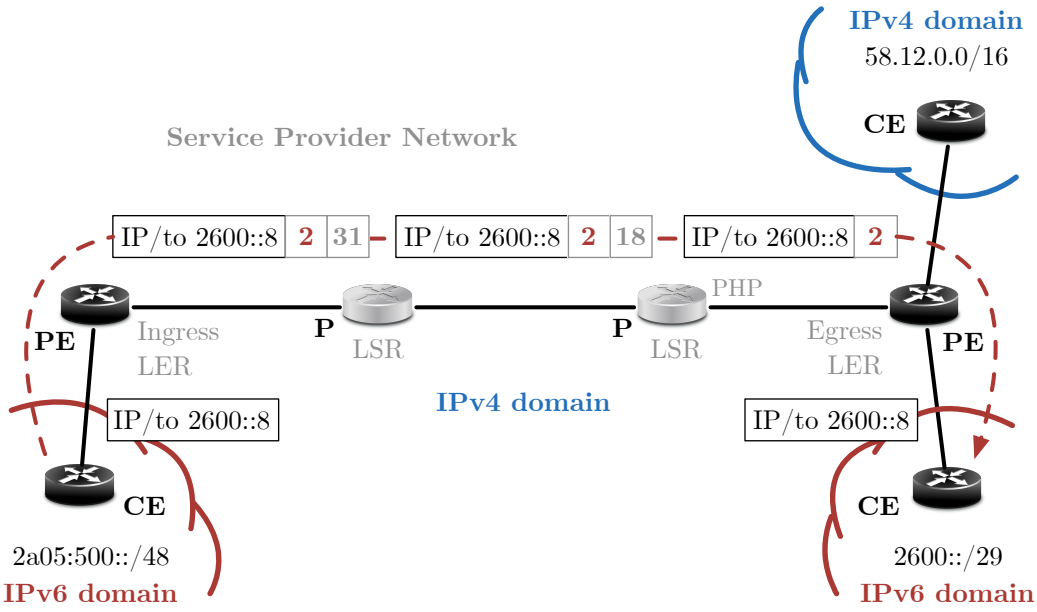


Figure 5.1: 6PE architecture. The PE routers are dual-stack, while the internal LSRs are IPv4-only devices.

(i) The top label is the transport label allowing to forward the packet between the ingress and egress LERs. It is assigned hop-by-hop [7, 12], and corresponds to the IPv4-signaled LSP.

(ii) The bottom label is assigned by MP-BGP, and advertised by MP-iBGP between the PE routers. It allows the IPv4 *penultimate hop (PH)* to apply PHP[4]. In this case, the router pops only the top of the stack, leaving the bottom label in the packet. It ensures that an IPv6 packet will not appear in an IPv4 core (which would occur if only one label was used), and signals directly to the egress LER that it is facing IPv6 traffic. Quoting RFC4798 [47], "This label advertised by the egress 6PE Router with MP-BGP *may* be an arbitrary label value, which identifies an IPv6 routing context or outgoing interface to send the packet to, or *may* be the IPv6 Explicit Null Label". This last label has a value of 2, as already mentioned in Section 1.5.5.

Note that today, now that the global IPv6 deployment is more common, 6PE is also interesting for dual-stack core LSRs with an IPv6 connectivity. Indeed, it allows to build LSPs for IPv6 without using an IPv6 label distribution protocol (LDP for IPv6 [9] was only finalized recently), and/or for sharing the same LSP for IPv4 and IPv6 traffic, reducing so the control plane complexity. We will see in the next section that this specific behavior is the most common in practice.

Finally, this chapter focuses on explicit MPLS tunnels, which can be fully revealed with `traceroute` (see Section 2.3.1). In the case of 6PE, if the TTL of a `traceroute` probe expires

[4]The LSP may be already used for native IPv4 traffic with PHP.

inside the IPv4 core, the LSR may be unable to send an ICMPv6 error message. The trace is then incomplete, and the non-responding hop is replaced by a star (*) in the output. However, if the router does not have any IPv6 connectivity, but is IPv6-aware, it may send an ICMPv6 message, using a so-called *IPv4-mapped IPv6* address[5] [79] as source address. The error message is then propagated towards the egress LER using MPLS, and is then forwarded based on IPv6 routing.

5.3 Evaluation

In this section we will study the deployment and use of MPLS in IPv6 networks based on `traceroute` data. After a description of the dataset, we will compare IPv4 and IPv6 MPLS tunnels according to the size and content of their label stacks. In the last part, we will analyze the specific case of COGENT, a large Tier-1 ISP having a particular use of 6PE.

5.3.1 Dataset

The IPv6 `paris-traceroute` data analyzed in this chapter was collected by CAIDA's Archipelago measurement platform. In the infrastructure, each monitor probes a single random destination in each announced IPv6 prefix (/48 or shorter) once every 48 hours. Some measurement points may also target the first address (i.e., ::1) in a prefix.

Table 5.1 provides raw statistics on the different probing campaigns performed every January 1st between 2009 and 2015. Note that the IP addresses of all the explicit MPLS tunnels identified in these sets of traces were mapped to their AS using Team Cymru [140]. The table shows a slow adoption of IPv6 between 2009 and 2013, compensated by a fast increase between 2014 and 2015. This observation confirms the conclusions of other studies [43, 51] conducted by the research community. The MPLS deployment in IPv6 networks follows that tendency, the peak being reached in 2014 and 2015.

In the following, we will focus on the data collected between January 1st, 2014 and August 1st, 2015. The considered dataset contains the first measurement snapshot of each month for that period of time, which represents a total of 20 measurement cycles.

Figures 5.2 and 5.3 give an overview of the MPLS deployment in IPv6. In particular, Figure 5.2 shows the raw number of traces crossing at least one MPLS tunnel. Their amount remains quite stable, even if the quantity of `traceroutes` increases over time. IPv6 traces involve also much less MPLS tunnels than IPv4 ones[6] (about 7-8% against at least 40% respectively). Note that the drops observed in early 2015 are due to a fewer number of active vantage points during this period in the measurement platform.

[5] An IPv4-mapped IPv6 address is an address that represents an IPv4 address in the IPv6 format. It is used to allow IPv6-enabled devices to deal with an IPv4 address as if it was an IPv6 one.

[6] An analysis of the number of IPv4 traces traversing at least one MPLS tunnel was performed in Section 4.4.1.

Year	Probing				Addresses			Tunnels	
	VPs	Traces	Prefixes	ASs	v6	v4 map'd v6	MPLS	Raw	Complete
2009	5	7,765	2,128	988	4,009	0	14	47	68%
2010	8	17,472	3,550	1,363	6,331	21	48	59	52%
2011	13	51,636	8,347	2,365	12,307	211	199	1,235	22%
2012	21	154,791	18,589	3,918	23,225	704	680	2,783	42%
2013	25	256,725	25,891	4,992	33,239	370	1,468	14,366	45%
2014	29	772,461	32,391	6,224	43,309	719	2,526	49,232	77%
2015	29	1,181,139	38,901	8,181	58,150	420	3,098	50,805	85%

Table 5.1: Raw IPv6 statistics over 7 years of data (January 1st of each year), where "VPs" (*Vantage Points*) gives the number of probing monitors, "Traces" the total number of traces that were collected, "Prefixes" the number of probed prefixes, "ASs" the amount of different ASs observed in the data, "Addresses" the number of pure IPv6 addresses, IPv4-mapped IPv6 addresses and addresses involved in MPLS IPv6 tunnels, and "Tunnels" the number of unique explicit MPLS tunnels identified in the traces (note that "Complete Tunnels" refers to tunnels for which all LSRs responded to `traceroute` probes).

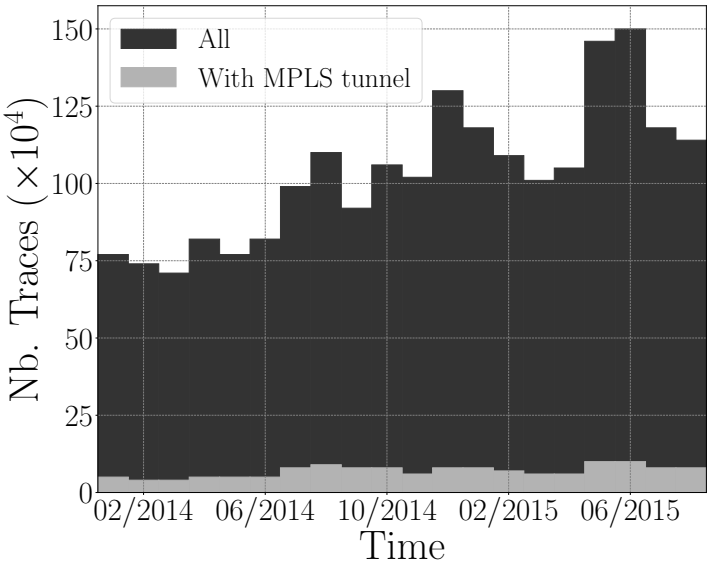


Figure 5.2: Raw number of IPv6 traces collected by CAIDA.

Figure 5.3 gives the MPLS tunnel length distribution for four measurement snapshots. Note that the ingress and egress LERs are not included in the metric[7], meaning a length of 1 corresponds to a tunnel with a single internal LSR. The figure shows that the tunnels comprise between 1 and 19 internal routers. More interestingly, their length seems to decrease over time, meaning that LSPs observed in 2015 are shorter than in 2014. The reason of this trend is AS174 (COGENT). It disappears from the dataset around October 2014, while it made use of long

[7]The length of a tunnel is defined as the length of its LSP, i.e. the number of internal LSRs.

tunnels[8]. In summary, the length distribution of IPv6 MPLS tunnels follows the observations made in IPv4 [53, 137], even if longer LSPs may be encountered.

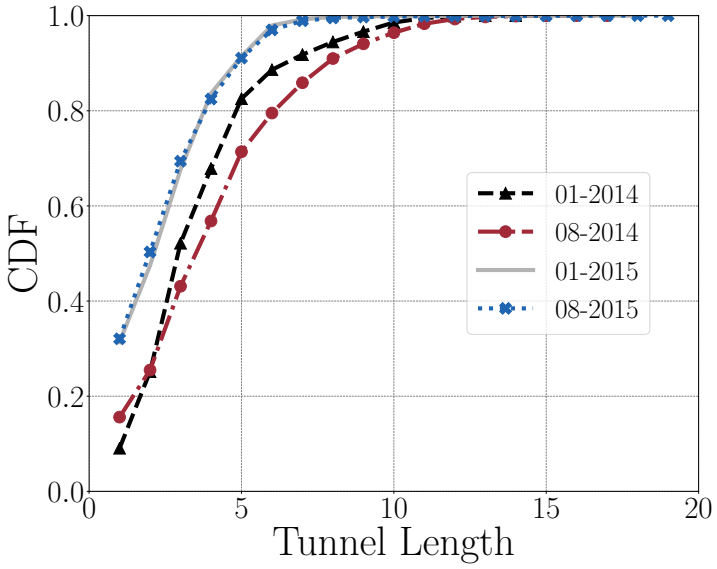


Figure 5.3: IPv6 MPLS tunnel length distribution.

5.3.2 Label Stack

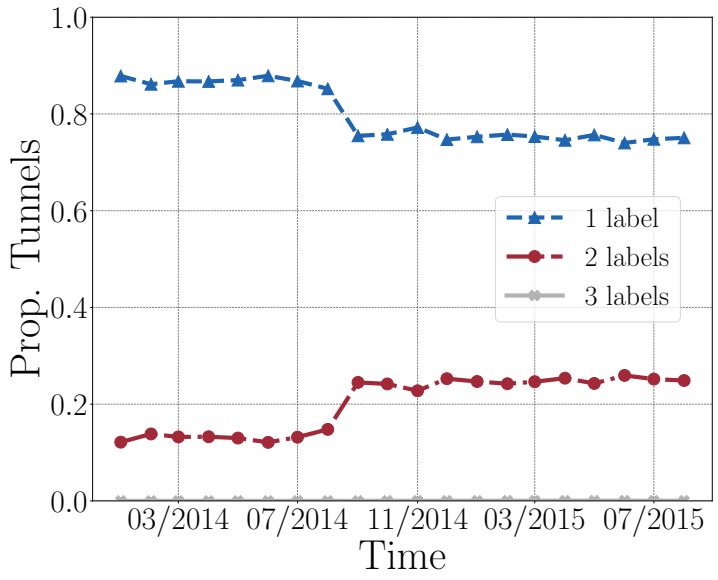
In this section, we will study the characteristics of IPv6 MPLS tunnels compared to IPv4 ones. In the first place, we will focus on the typical LSE stack size in both data planes, meaning the number of MPLS labels appearing in the packets at each single LSR.

The followed methodology is quite simple, as each tunnel is mapped to the largest stack size observed in its LSP. Short tunnels are then associated to their most likely use. For instance, an IPv4 tunnel made of three LSRs with the sequence 1,2,1 (in terms of LSE stack sizes) is probably deployed for VPN purposes, and its LSP is considered as a 2-label LSP. Note that, in such a case, the bottom label is never modified in the MPLS network, as already explained in Section 1.6.1.

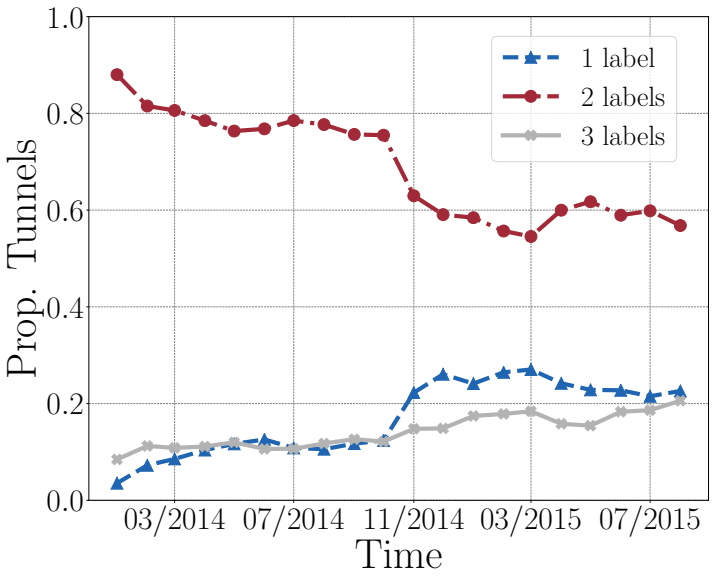
Figure 5.4 shows the stack size distribution over time, between January 1st, 2014 and August 1st, 2015. Globally speaking, IPv4 (Figure 5.4(a)) and IPv6 (Figure 5.4(b)) MPLS tunnels exhibit different behaviors. Indeed, under IPv4, the majority of LSPs (around 80%) have only a single LSE in their stack. This result is aligned with the observations of Sommers et al. [137]. In opposition, most IPv6 tunnels (also around 80% during the first half of the considered period) use at least two labels[9]. This result may appear really surprising since there is no obvious reason justifying a more extensive use of VPNs in IPv6 than in IPv4.

[8]The impact of AS174 in IPv6 has already been discussed in the past [69, 94].

[9]The drop around October 2014 in IPv6 is due to AS174 (COGENT), a heavy user of 2-LSE stacks that got rid of MPLS. This case will be discussed in detail in Section 5.3.3.



(a) IPv4 MPLS tunnels.



(b) IPv6 MPLS tunnels.

Figure 5.4: LSE stack size distribution over time.

Figure 5.5 deeper investigates the content of the LSE stacks in IPv6. In particular, it shows the values of the bottom labels for the tunnels with at least one LSR having 2 or more LSEs in its stack. As explained in Section 5.2, a value of 2 suggests a 6PE architecture where core LSRs are dual-stack capable[10]. A clear shift can be observed around October 2014. Prior this date, MPLS routers use 2 as bottom label almost as often as other values. After October 2014, things change radically. The majority of LSPs (more than 80%) use the bottom label 2, suggesting that the use of 6PE is prevalent.

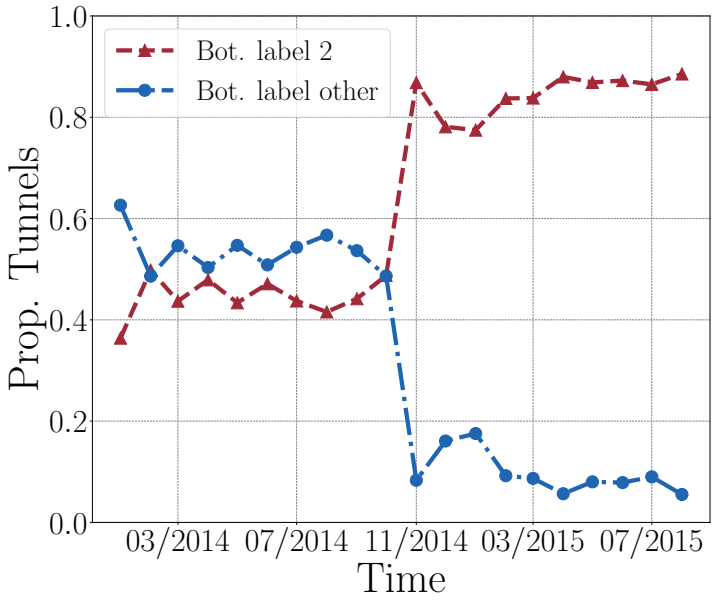


Figure 5.5: Distribution of the bottom label value in IPv6 LSE stacks.

To go further in the analysis, the bottom LSE of the 6PE tunnels (i.e., label 2) was removed, and the series of remaining labels were compared with the series obtained from IPv4 MPLS tunnels identified in traces collected during the same period[11]. A match was found in more than 40% of the cases. The same IPv4 LSP is then used for IPv4 and IPv6 traffic, reinforcing so the assumption of the 6PE architecture.

The radical change depicted in Figure 5.5 is very surprising at first glance. To explain it, each AS encountered in the dataset around this date was analyzed. Before October 2014, around 50% of the tunnels belong to AS174 (COGENT). In November 2014, they all disappear from the traces, while the AS remains visible through numerous non-MPLS IP addresses. Almost all tunnels belonging to this prominent Tier-1 network have a 2-label stack, but they never use 2 as bottom label, which explains the shift observed in Figure 5.5. The specific label stacks used by COGENT will be studied in detail in Section 5.3.3, as they are almost only specific to this AS.

[10] If the core was only IPv4-capable, the different LSRs would not have responded to IPv6 `traceroute` probes.
[11] These IPv4 MPLS traces were also downloaded from the Archipelago platform.

Figure 5.6 depicts the architecture of the network core in case of dual-stack 6PE (i.e., bottom label 2). A tiny proportion of tunnels map IPv4 addresses into IPv6 ones (black region in the figure). It happens when the core LSRs are IPv6-aware (i.e., they are dual-stack), but are not assigned any public IPv6 address. Therefore, in order to generate their ICMPv6 `time exceeded` messages, they map their IPv4 address into a "fake" IPv6 one. Most dual-stack 6PE tunnels observed in the dataset have an IPv6 core, as their LSRs responded with a public IPv6 address. It means that the different LSPs, generally built with LDPv4, and attached to a given IPv4 loopback address of an egress LER, are used for both IPv4 and IPv6 traffic. In this case, the series of top labels are identical in both data planes, while the bottom label 2 indicates to the egress LER when the traffic is made of IPv6 packets rather than IPv4 ones.

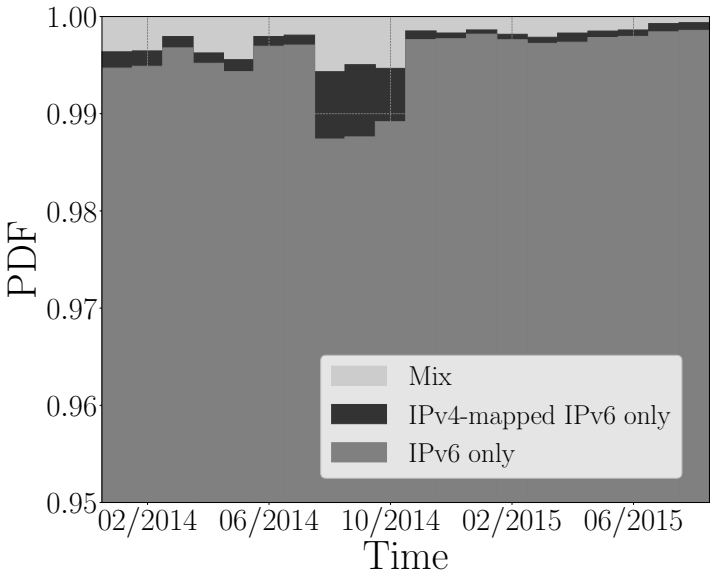


Figure 5.6: 6PE core architecture.

Note that, originally, 6PE was used to ensure a connectivity between IPv6 islands in a pure IPv4 core. In this case, the IPv4 LSRs are not IPv6-aware, and do not respond to the IPv6 probes of `traceroute`. The traces in the dataset are then incomplete, as a sequence of stars (*) appears between the ingress and egress LERs. Unfortunately, it is impossible to differentiate such a behavior from IPv6 nodes that simply did not respond to probes. The proportion of 6PE tunnels is therefore underestimated in this study.

5.3.3 The Cogent Case

As already stated in the previous section, AS174 (COGENT) is particularly interesting, and quite intriguing. First, it is one of the largest Tier-1 networks, and has a very prominent position in the dataset. According to CAIDA, it has the third highest AS rank [28]. Then, its IPv6 MPLS behavior is completely different from the other ASs identified in the collected traces. Indeed,

when COGENT's MPLS tunnels disappear from the dataset around October 2014, the global proportion of LSPs with bottom label 2 rises sharply, as shown in Figure 5.5.

Two reasons may explain the decrease of the number of MPLS tunnels in the COGENT network after this date. Indeed, the operator may have modified the configuration of his routers, or, more simply, may have stopped using MPLS. In the first case, disabling the IP TTL propagation at the ingress LERs is enough to turn all explicit tunnels into invisible (or opaque) tunnels, as already explained in Section 2.3. Note that a similar phenomenon touching LEVEL3's IPv4 tunnels was discussed in Section 4.4.4. To validate, or invalidate this hypothesis, a subset of MPLS tunnels identified before October 2014 was selected. The different sequences of IP addresses between the ingress and egress LERs were then compared with traces collected after this date. As a result, the same paths were observed in both sets, the only difference being that MPLS labels disappeared after October 2014. We can then conclude that the AS disabled MPLS in its IPv6 network, as it already did for IPv4 two years before. Note that a network administrator working for COGENT, and contacted for this study, confirmed this result.

A second and more interesting fact revealed by Figure 5.5 is that, although most of COGENT's LSPs have a stack size greater or equal to 2, the bottom label 2 is never used, in opposition to other ASs (see Section 5.3.2). Note that RFC4798 [47] does not force the use of label 2 as bottom label for the 6PE architecture, and specifies that an egress router may associate any label to an IPv6 prefix, and announce it to its iBGP peers. Therefore, a 6PE implementation could select any arbitrary value for all tunnels, or choose a different label for each prefix, or set of prefixes.

An analysis of COGENT's LSE stacks revealed that the bottom label is not fixed, but varies greatly. So, the AS does not simply use another arbitrary value than 2. In fact, numerous different bottom labels can be found in LSPs connecting the same <*ingress LER, egress LER*> pair. For instance, 38 distinct bottom labels can be observed for a given pair in the dataset. In theory, these labels could be linked to 38 VPNs, or more probably, the egress LER could use a distinct value for each (group of) IPv6 prefix. Fortunately, the COGENT's network administrator eliminated the VPN case. Indeed, nothing can distinguish the VPN and the 6PE architectures in terms of measurements, the general principle of using a bottom label being the same. Therefore, the AS only used the 6PE technolgy before shutting down MPLS for IPv6 in October 2014.

One purpose of using distinct bottom labels in 6PE tunnels may be load sharing. In a network using *Equal-Cost Multi-Path (ECMP)*, packets arriving at a router with two equal-cost paths to their destination are distributed along these routes according to a hash based on their header. In a network using both MPLS and ECMP, LDP makes use of multiple routes, and builds several LSPs between the same pair of LSRs. Different proposals also describe extensions to RSVP-TE in order to take into account multiple available paths [91]. In addition, several mechanisms are suggested to improve load balancing in MPLS networks [90]. In the case of COGENT, it is clear that the core network performs load sharing. For example, reconsidering the <*ingress LER, egress LER*> pair with 38 distinct bottom labels, if the bottom label is removed, 8 distinct LSPs

persist between the two routers (considering IP addresses and top labels).

With IPv4 packets, the hash function considers at least the source and destination IP addresses in order to guarantee the same route for all packets of the same flow (avoiding so ordering issues with TCP). The same can be done with IPv6 packets, but at a higher cost due to the length of IPv6 addresses. Moreover routers in the core of a 6PE tunnel may be totally IPv6-unaware. In this situation, using the bottom label to split the load makes sense[12]. Note that the use of many ECMP routes generates multiple 6PE bottom labels. Considering also the top label values, a large number of distinct LSPs may arise in the network when taking into account the full label stack. It partially explains why COGENT is so present in terms of unique IPv6 MPLS tunnels in the dataset, besides its dominant size.

To investigate further, and retrieve the root cause of this variety of label stacks, the *Label Pattern Recognition (LPR)* algorithm, presented in Chapter 4, was applied on the top labels of COGENT's IPv6 MPLS tunnels. It allowed to quantify the use of LDP (for IGP-BGP scalability purposes) and/or RSVP-TE (for traffic engineering). The classification of the different <*ingress LER, egress LER*> pairs is available in Table 5.2. The results show that the top label is mostly generated by LDP (Mono-FEC line in Table 5.2), which confirms the previous analysis of the case with 38 distinct LSPs. Therefore, the bottom label is probably assigned by the egress LER on a per-IPv6-prefix basis, using a variant of 6PE, in order to make a more efficient use of ECMP, while the top label (i.e., the LSP itself) is generated by LDP for IPv4.

Class	**Snapshot**		
	09/2014	10/2014	11/2014
Mono-LSP	23.1%	30.8%	0%
Multi-FEC	3.4%	2.7%	0%
Mono-FEC	58.6%	52.3%	0%
Unclassified	14.9%	14.2%	0%

Table 5.2: Classification of COGENT's <*ingress LER, egress LER*> pairs by LPR, for three snapshots in 2014.

5.4 Conclusion

The last years have seen an increasing adoption of IPv6. With the recent IPv4 depletion, this increase is going faster, and more and more new IPv6 networks are expected in the future. In this chapter, we focused on the deployment and use of MPLS in IPv6 domains based on `paris-traceroute` data collected by CAIDA.

Our first observations showed that MPLS is used differently in IPv4 and IPv6. In particular, with an analysis of the label stacks, we discovered that under IPv4, LSPs with more than one label are not that frequent, while they are the norm under IPv6. However, we showed that this

[12]This practice is mentioned in CISCO's documentation [36].

difference is not due to an extensive use of VPNs, but is linked to 6PE tunnels built using an IPv4 signaling protocol (in particular LDP for IPv4). This mechanism allows to deploy MPLS for IPv6 across a network where some routers are not dual-stack, or where LDPv6 is not yet available (this version was released quite recently). Therefore, 6PE can be seen as a transition technology. Finally, the special case of COGENT's network also shed some light on the use of ECMP in conjunction with MPLS. We discovered that this AS uses a specific form of 6PE to ease the way IPv6 routers select their outgoing interfaces.

PART III

INVISIBLE MPLS TUNNELS REVELATION

CHAPTER 6

REVEALING PHP INVISIBLE MPLS TUNNELS

For years, Internet topology research has been conducted through active measurement. For instance, CAIDA builds router level topologies on top of IP-level traces obtained with `traceroute`. The resulting graphs contain a significant amount of nodes with a very large degree[1] [116], often exceeding the actual number of interfaces of a router. Although this property may result from inaccurate alias resolution[2] [88], MPLS clouds made of invisible tunnels are probably the main cause. Indeed, as explained in Chapter 2, MPLS devices can be configured to hide internal IP hops from `traceroute`. Consequently, an entry point of an MPLS network appears as the neighbor of all exit points, and the whole Layer-3 network turns into a dense mesh of high degree nodes. This chapter tackles three problems [146]: (i) the revelation of IP hops hidden by MPLS tunnels, (ii) the underestimation of MPLS deployment, and (iii), the overestimation of high degree nodes. More specifically, it presents new measurement techniques able to reveal the presence and content of invisible MPLS tunnels that perform *Penultimate Hop Popping (PHP)*[3]. After a description of these techniques, we will see how they could be assessed through emulation and cross-validation on a large-scale measurement campaign targeting suspicious networks. Finally, based on the collected dataset, we will study and improve the inference of basic graph properties impacted by invisible MPLS tunnels.

6.1 Introduction

Since the end of the nineties, Internet topology discovery has been extensively investigated. Indeed, numerous analyses [52, 76] describe various types of connectivity structures and repre-

[1] The degree of a node is defined as the number of neighboring nodes to which it is directly connected.
[2] Alias resolution is the process of determining if IP addresses belong to the same network device.
[3] The identification and revelation of other types of invisible tunnels will be addressed in Chapter 7.

sentations of the Internet architecture. In particular, inferring the router-level topology of IP networks is an important concern, especially for studying routing characteristics. These router-level maps are obtained based on `traceroute` data, by grouping together IP addresses belonging to the same device. This process is called *alias resolution* [88]. Inferring the architecture of an *Autonomous System (AS)* is also crucial for analyzing the performance of routing protocols. Using random graph models rather than realistic networking topologies may result in biased or even wrong conclusions. For example, the performance of fast-rerouting schemes or multi-path transport protocols strongly depends on the underlying topology.

Typically, router-level topologies are undirected graphs built upon IP-level traces obtained with `traceroute`. These graphs can be statistically analyzed to study the evolution and structure of a network [116]. In particular, the node degree distribution fascinates the research community, especially since the seminal paper of Faloutsos et al. [61] that highlighted its power-law[4] shape. However, inferred topologies may contain a significant amount of nodes with a very large degree, often exceeding the actual number of interfaces of a router. As an example, Figure 6.1 illustrates the degree distribution of the nodes in CAIDA's ITDK dataset[5] [29] collected in March 2016. We can observe a large amount of nodes having a very high degree.

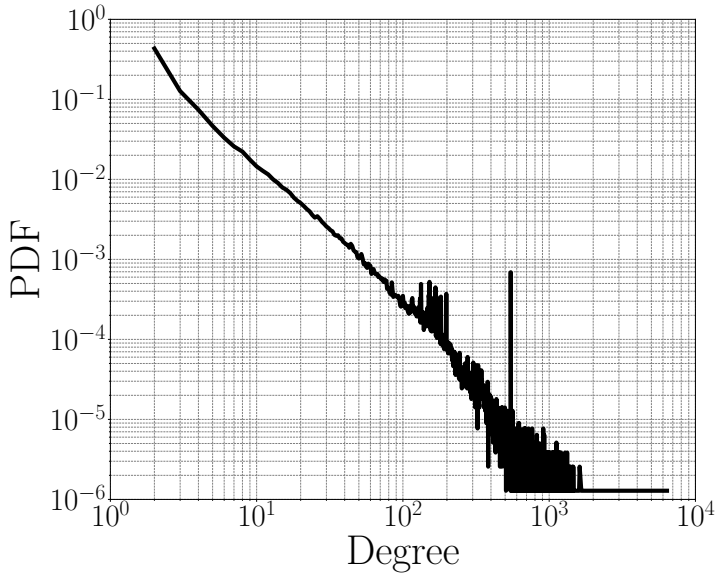


Figure 6.1: Node degree distribution in CAIDA's ITDK dataset

The quantity of *High-Degree Nodes (HDNs)* may be explained by several factors. First, a `traceroute` campaign conducted from a limited number of monitors tends to induce a subgraph in which the inferred node degree distribution follows a power law, even when it is not necessary the case [93]. Clauset and Moore [39] have since analytically demonstrated that such a phenomenon

[4]"A power law is a relationship in which a relative change in one quantity gives rise to a proportional relative change in the other quantity, independent of the initial size of those quantities" [17].

[5]This dataset provides router level topologies.

is to be expected for the specific case of Erdös-Rényi random graphs [60]. Second, others [107] have stated that HDNs can emerge from Layer-2 clouds (such as Ethernet switches). In these networks, Layer-2 devices interconnect a large number of Layer-3 routers, themselves being also involved in multiple Layer-2 interconnections. Such a situation induces nodes with very high degrees when analyzing the Layer-3 graph with `traceroute`.

In this chapter, we will investigate another reason of HDNs in the Internet graph: MPLS clouds hiding their content to `traceroute` measurements. As already stated in the previous chapters, MPLS is largely deployed by operators. It is also confirmed by a survey carried out as part of this thesis between August 28th, 2017 and September 12th, 2017. Out of the data collected through direct contacts with network operators and the NANOG community (50 answers, from Stub ISPs to Tier-1 networks), 87% of the surveyed ASs make use of MPLS.

Unfortunately, ISPs may want to hide the structure and the configuration of their internal MPLS network. For example, most of the time, providers prefer to simplify the routing view of their customers for VPN services[6]. In this case, the customers see their different sites as directly connected by the provider's border routers, and the details of the internal architecture are not shown. As already discussed in Chapter 2, disabling the TTL propagation is enough to turn an MPLS tunnel into an *invisible* tunnel dissimulating its internal LSRs to `traceroute`. In this way, operators can prevent competing networks to imitate their finely tuned calibration, and attackers to get knowledge of their internal organization [68].

Consequently, the data obtained by researchers from `traceroute` measurements is incomplete, and the resulting Internet maps are potentially biased. Indeed, as the content of the tunnel is hidden, a direct and false link is inferred between its entry and exit points. Further, an ingress LER of an MPLS network appears as the direct neighbor of all egress routers. Therefore, the whole Layer-3 network turns into a dense mesh of high-degree nodes [152]. As a result:

(i) Current basic `traceroute` campaigns cause the inference of false router-level links between two edge routers separated by an invisible tunnel.

(ii) The deployment of MPLS on the Internet may be underestimated, as internal IP links are not taken into account.

(iii) The node degree distribution, and other graph properties, such as density[7] or clustering coefficient[8], may be shifted to higher values.

Another reason for identifying invisible MPLS tunnels is to better capture network delay anomalies [65]. Indeed, as some hops are hidden in `traceroute` outputs, the delay between the entry and exit points of a tunnel might appear as being artificially high, possibly leading to wrong conclusions when tracking connectivity issues.

[6]The VPN architecture is explained in Section 1.6.1.

[7]The density of a graph is defined as the ratio of its number of edges to its maximum possible number of edges.

[8]The clustering coefficient reflects the degree to which nodes in a graph tend to cluster together [71].

The aim of this chapter is to fix those issues by suggesting new probing mechanisms and analyses that can be run when exploiting IP-level traces. First, after additional information on the TTL processing inside MPLS tunnels (Section 6.2), we will update the taxonomy presented in Chapter 2 for invisible tunnels that perform *Penultimate Hop Popping (PHP)* (Section 6.3).

Then, we will focus on new active measurement techniques (Section 6.4). Using `traceroute` and TTL-based estimations, they are able to signal the presence of PHP invisible tunnels, and, in the best cases, reveal their content. Note that MPLS tunnels with PHP are the most deployed in the Internet. Indeed, most manufacturers enable PHP by default on their equipment. Moreover, *Ultimate Hop Popping (UHP)*[9] is generally used in conjunction with sophisticated traffic engineering solutions, which do not reflect the most common situation. This statement is confirmed by the survey, as only 10% of the ASs enable UHP. All the techniques described in this chapter have been assessed through emulation testbeds with GNS3, a software running actual router OSs in virtualized routers [70], and through cross-validation. We will see that the content of invisible MPLS tunnels can be revealed in roughly 86% of the cases.

Afterwards, we will analyze a specific dataset collected with the different measurement techniques, and show that the MPLS knowledge of the research community can be improved (Sections 6.5 to 6.7). We will also get an insight on standard and common practices of ISPs.

Finally, and in order to illustrate the contribution of this chapter, we will improve classical Internet topology models by correcting the biases in terms of node degree, route length distribution, and graph density (Section 6.8).

6.2 MPLS TTL Processing

At the end of an MPLS tunnel, the *Exit Hop (EH)*, meaning the egress LER, or the *Penultimate Hop (PH)* if PHP is enabled, decrements the LSE TTL. If the packet must not be dropped, the router has to determine the new IP TTL value before popping the label stack. On the one hand, if `no-ttl-propagate` is used at the ingress LER, the EH should not modify the IP TTL field in the packet's header. On the other hand, the router should copy the LSE TTL value if the `ttl-propagate` option is enabled. The operation performed at the EH depends then on the configuration at the ingress LER. In order to synchronize both ends of the tunnel without any message exchange, two mechanisms can be used for choosing the value of the IP TTL at the tunnel exit:

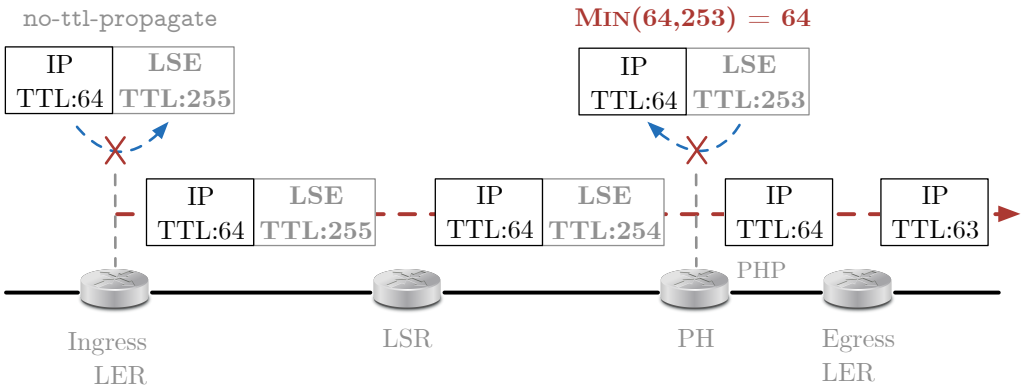
(i) Selecting the smallest value out of the IP and LSE TTLs. In other words, it consists in performing MIN(IP TTL, LSE TTL). This solution is implemented by CISCO with PHP configurations [48], and by multiple JUNIPER devices.

(ii) Assuming that the configurations (`ttl-propagate` or not) of the ingress LER and the EH are the same. So, for example, if the EH is configured to not propagate the IP TTL, it will

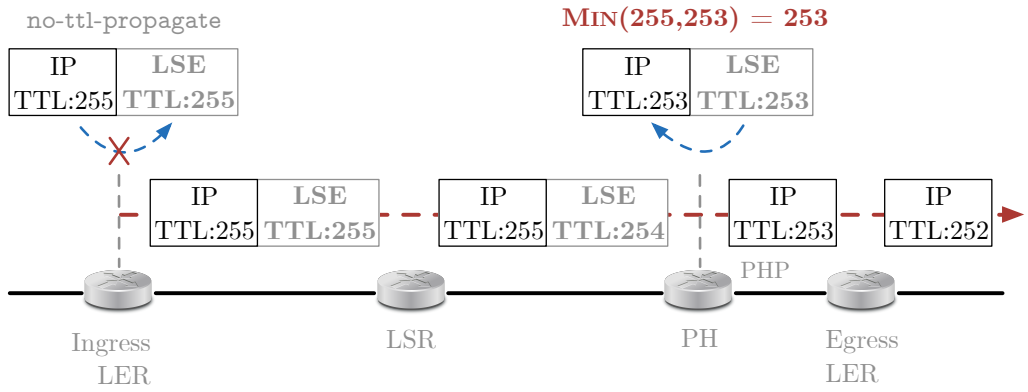
[9]The functioning of UHP and PHP is described in Section 1.5.5.

consider that it is also the case for the ingress LER, and it will not copy the LSE TTL in the IP TTL field. This solution implemented by JUNIPER in some Junos OSs, such as Olive, and also by CISCO with UHP.

Applying MIN(IP TTL, LSE TTL) is the best option, because it correctly supports heterogeneous `ttl-propagate` configurations in any case, and at the same time, mitigates forwarding loops in a stateless manner, without any signalization. It is illustrated in Figure 6.2 for a packet generated by the ingress LER of an invisible tunnel performing PHP (the TTL propagation is disabled). Note that in the case of an initial IP TTL of 255 (bottom of the figure), the MIN behavior allows the PH to behave as if the `ttl-propagate` option was enabled at the tunnel entry.



(a) The MIN operation selects the IP TTL at the PH.



(b) The MIN operation selects the LSE TTL at the PH.

Figure 6.2: MIN behavior for a packet generated by the ingress LER of an invisible tunnel performing PHP. The TTL propagation is not enabled at the ingress LER.

In practice, the second mechanism, i.e., assuming that the configuration of the TTL propagation is identical for each MPLS router in the tunnel, may lead to side effects in `traceroute` [143].

Indeed, in an heterogeneous configuration where `ttl-propagate` and `no-ttl-propagate` are used respectively at the entry and exit of the tunnel, the router that pops the label stack does not copy the LSE TTL in the IP TTL field. Consequently, even if the LSRs reveal themselves, the IP routers directly following the MPLS tunnel will be hidden in the `traceroute` output. This *jump* effect applies to as many hops as the LSP length, as shown in Figure 6.3. As heterogeneous cases are not in the scope of this document, we will always consider that MPLS tunnels are configured homogeneously, except when explicitly stated.

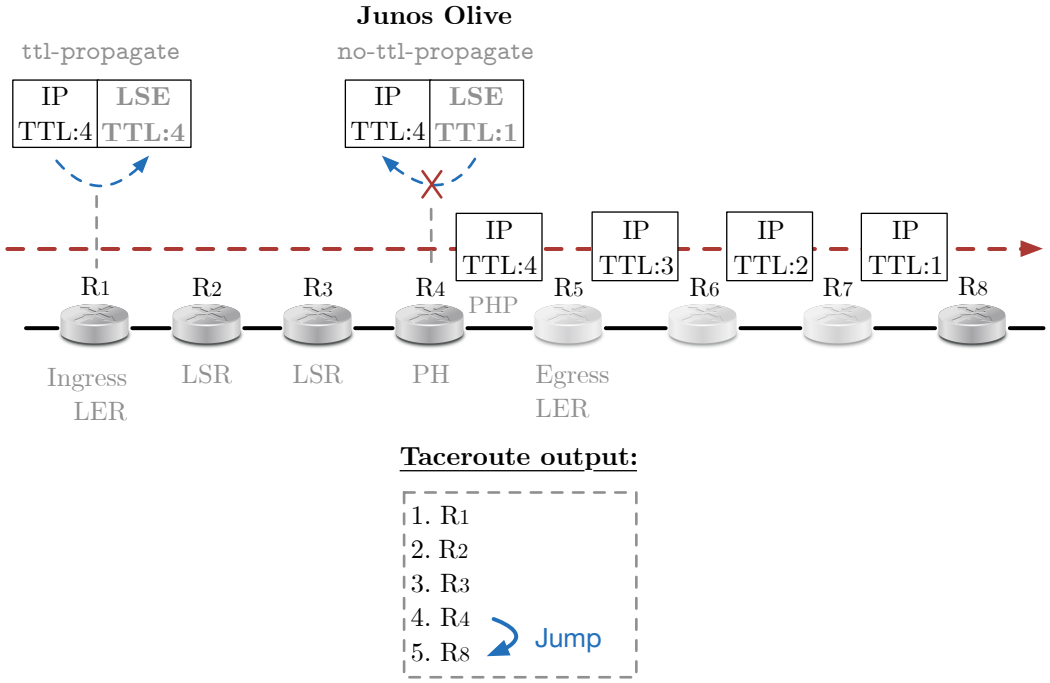


Figure 6.3: TTL selection at the PH based on the local configuration in a non-homogeneously configured MPLS tunnel. A *jump* as long as the LSP length is observed after the PH in the `traceroute` output.

6.3 PHP Invisible MPLS Tunnels

In the remainder of this chapter, we will focus on *PHP invisible MPLS tunnels*, i.e., tunnels performing PHP that are completely obscured from `traceroute`. According to the model of Donnet et al. [53] presented in Section 2.3.4, the ingress LER does not propagate the IP TTL, and the PH replies to a probe with an ICMP `time exceeded` message that does not contain any MPLS information, as RFC4950 is not implemented by the LSR. However, this model is not fully correct. Indeed, when PHP is enabled, the PH is in charge of converting the MPLS data packet into a standard IP one, as expected. However, in practice, this LSR never sends back any ICMP message (no matter RFC4950 is supported or not), because it does not decrement the IP

TTL[10]. As a result, the packet is considered as an IP one, and is simply pushed to the egress LER. Hence, all the MPLS routers inside the tunnel are hidden, PH included, and do not appear in the `traceroute` output. An update of the model presented in Section 2.3.4 is available in Figure 6.4. Note that in the survey, surprisingly, 48% of the operators make use of the `no-ttl-propagate` option.

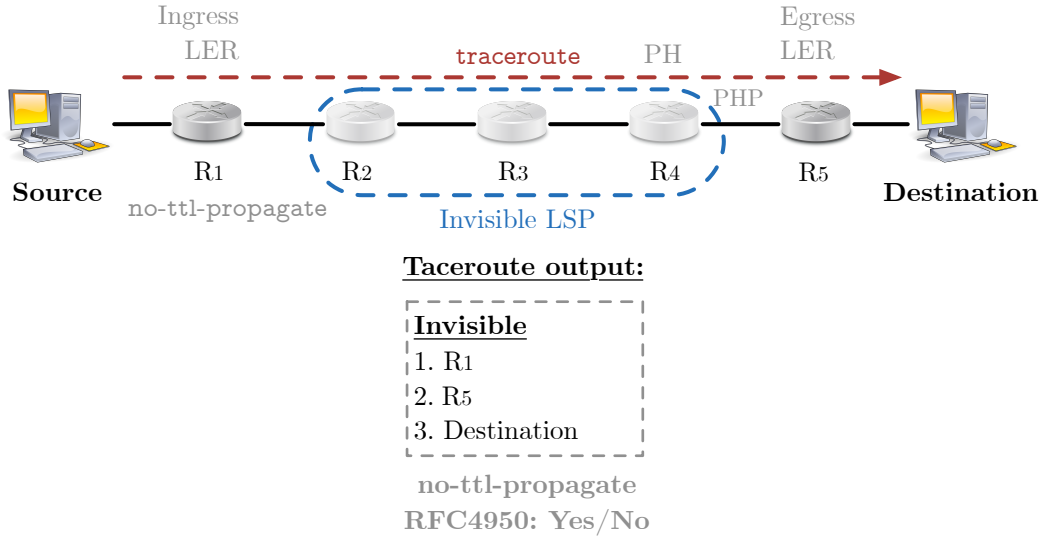


Figure 6.4: Update of the taxonomy presented in Section 2.3.4 in the case of invisible PHP MPLS tunnels. None of the internal hops appears in the `traceroute` output, penultimate hop (R_4) included.

6.4 Discovering PHP Invisible MPLS Tunnels

This section describes four complementary techniques for revealing the content, or at least identifying the presence, of PHP invisible MPLS tunnels. They rely on `traceroute` and `ping`, and can be classified into two categories. First, *Return / Forward Path Length Asymmetry (RFPLA)* and *Return Tunnel Length (RTL)* are only able to provide more or less high-level information about invisible MPLS tunnels. They are based on the idea that when the probes sent by the monitor cross an MPLS cloud, their corresponding responses are also likely to cross it again in the other direction. In the following, we will call *forward tunnel* the MPLS tunnel on the path from the source to the destination (i.e., followed by the probes), and *return tunnel* the one on the route from the destination to the source (i.e., followed by the replies to the probes)[11]. Thereby,

[10] As explained in the previous section, the IP TTL value is either not modified, or replaced with the LSE TTL value. Moreover, the LSE TTL does not expire as it is initialized to 255 at the ingress LER.

[11] Note that in practice, the forward and return tunnels may be similar (the forward ingress LER becomes the return egress LER, and inversely), or completely different.

while RFPLA gives an estimation of the number of internal LSRs in the return MPLS tunnel, RTL is able to determine their exact quantity. Those methods signal only the presence of an invisible MPLS network, and are not able to expose the internal IP hops hidden by the tunnels. Second, *Direct Path Revelation (DPR)* and *Backward-Recursive Path Revelation (BRPR)* allow to explicitly reveal the content of the obfuscated tunnels (the hidden internal LSRs), either with a single additional measurement shot, or hop-by-hop with a recursive probing process.

Combining those four techniques allows to capture a majority of MPLS use cases, meaning typical network MPLS/IGP/BGP configurations, and the standard behaviors of CISCO and JUNIPER devices, where LDP[12] announces respectively a label for all the prefixes in the IGP routing table, or for the loopback address only. Table 6.1 summarizes the scope of the four measurement techniques for different MPLS configurations.

LDP advertising	**Target** (`traceroute`)	**TTL propagation policy**	
		`ttl-propagate`	`no-ttl-propagate`
All IGP prefixes	external	visible LSP	invisible LSP RFPLA and RTL positive
	internal	visible LSP	only PH visible w/o label (BRPR) RFPLA and RTL positive
Loopback addresses only	external	visible LSP	invisible LSP RFPLA and RTL positive
	internal	visible IP route	visible IP route w/o labels (DPR) RTL positive

Table 6.1: Visibility effects of basic MPLS configurations according to the label advertisement policy. The `traceroute` target is either "external" if it is located outside the AS deploying the invisible MPLS tunnel, or "internal" if it is inside. The TTL policy (enabling the IP TTL propagation or not) is assumed to be similar on both LERs. The last column specifies if RFPLA and RTL signal the presence of the tunnel, and if DPR and BRPR can reveal its content. Note that RTL only works with JUNIPER egress LERs (see Section 6.4.1), and that the considered tunnels perform PHP.

6.4.1 Inferring the Length of Tunnels

The path length is a important metric often used when performing measurements in computer networks. In order to be consistent with the definition of tunnel length (i.e., the number of LSRs in the LSP), in the remainder of this document, the length of a path is defined as the number of intermediate nodes (i.e. routers) between its source and destination.

The two first techniques (RFPLA and RTL) are based on the same principles. They exploit the lengths of the paths followed by the `traceroute` and `ping` probes sent to the forward egress LER,

[12]In the survey, 50% of the operators only deploy LDP, 8% use RSVP-TE alone, and the remainder runs both protocols in conjunction.

and their corresponding responses. Only 3 lengths, represented in Figure 6.5, are considered: L_R^T (path taken by the `time exceeded` message), L_R^P (path taken by the `echo reply` message), and L_F^T (forward path taken by the `traceroute` probe). More precisely, RTL is the difference between the lengths of the return paths for the `time exceeded` and `echo reply` messages, (i.e., L_R^T- L_R^P), while RFPLA is the difference between the return and forward path lengths for `traceroute` (i.e., L_R^T- L_F^T).

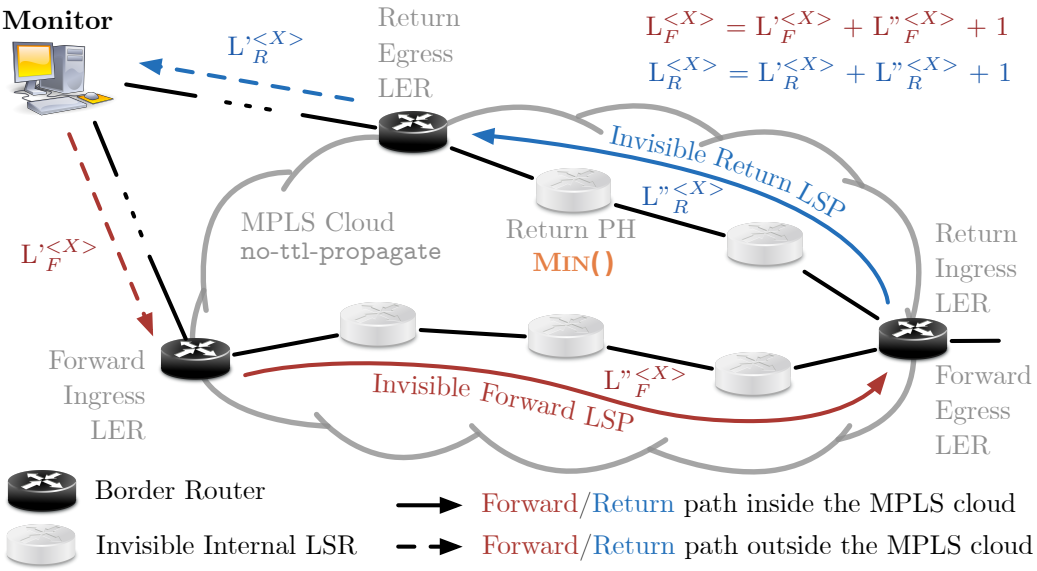


Figure 6.5: Illustration of the path lengths used by RFPLA and RTL. Notations L' and L" refer respectively to the path lengths outside and inside the MPLS cloud. On their side, the subscripts F and R signal respectively the forward (from the monitor to the forward egress LER) and return (from the return ingress LER to the monitor) paths. Finally, the superscript $< X >$ may represent either `traceroute` (T), or `ping` (P). For example, $L"_R^T$ identifies the length of the return path in the MPLS cloud, taken by the ICMP `time exceeded` message sent by the return egress LER in response to a `traceroute` probe. Consequently, $L_F^T = L'_F^T + L"_F^T + 1$. Note that the last term (+ 1) allows to take into account the return egress LER.

In the absence of invisible tunnels, RFPLA and RTL should have a value equal to or close to 0. Indeed, in this case, $L"_F^T = L"_R^T = L"_R^P = 0$ (see Figure 6.5), and it comes:

$$\begin{aligned} \text{RFPLA} &= L_R^T - L_F^T \\ &= (L'_R^T + L"_R^T + 1) - (L'_F^T + L"_F^T + 1) \\ &= L'_R^T - L'_F^T \end{aligned} \tag{6.1}$$

$$\begin{aligned} \text{RTL} &= L_R^T - L_R^P \\ &= (L'^T_R + L''^T_R + 1) - (L'^P_R + L''^P_R + 1) \\ &= L'^T_R - L'^P_R \\ &= 0 \end{aligned} \tag{6.2}$$

Equation 6.1 shows that RFPLA is subject to BGP path asymmetry. Indeed, the difference is close to 0 only if the forward and return paths are similar outside the domain (i.e. if BGP does not interfere). On its side, RTL is not affected by this problem (see Equation 6.2), as the `time exceeded` and `echo reply` messages follow the same physical route to the monitor, and therefore use the same BGP return path. Note however that it may produce some false alarms in networks deploying ECMP.

If the path to the destination crosses an invisible MPLS cloud, RTL and RFPLA should significantly deviate from 0. Indeed, in this case, they respectively infer the exact and approximate return LSP length, exploiting the result of the MIN(IP TTL, LSE TTL) operation applied by the PH at the exit of the return tunnel.

Both techniques depend on the behavior of the forward egress LER. When it receives a probe, it generates an ICMP message that must be sent to the monitor. This LER, that becomes the new ingress LER on the return path, must initialize the LSE and IP TTLs of the new packet. As the MPLS cloud is invisible, the device is configured to not propagate the IP TTL, and the LSE TTL is always set to 255. Referring to Chapter 3, and more specifically to Table 3.1 and Figure 3.6 of this thesis, operators deploy mostly CISCO and JUNIPER equipment in their network. Then, most of the time, the IP TTL is initialized to 255, except for ICMP `echo reply` messages if the LER is a JUNIPER device. In this case, the value 64 is used.

Consequently, when the return ingress LER sets both TTLs to 255, for example for an ICMP `time exceeded` message, the return PH necessarily selects the LSE TTL when it applies the MIN operation at the exit of the tunnel. Indeed, only the LSE TTL was decremented inside the MPLS network, and the IP TTL remained to 255. Therefore, the tunnel behaves as if the IP TTL was propagated by the ingress LER (see Figure 6.2(b)). If we assume that forward and return paths outside the cloud are similar (i.e. $L'^T_R \approx L'^T_F$), applying RFPLA on the forward egress LER gives:

$$\begin{aligned} \text{RFPLA} &= L_R^T - L_F^T \\ &= (L'^T_R + L''^T_R + 1) - (L'^T_F + L''^T_F + 1) \\ &\approx L''^T_R - L''^T_F \\ &\approx L''^T_R \end{aligned} \tag{6.3}$$

Indeed, the MPLS tunnel is invisible on the forward path ($L''^T_F = 0$), while the length of the one on the return path is taken into account in the IP TTL of the ICMP message received by the monitor, thanks to the MIN operation ($L''^T_R \neq 0$). Note that in practice, L_F^T can be obtained subtracting 1 to

the IP TTL used by `traceroute` for the probe dropped at the forward egress LER, while L_R^T can be easily computed as 255 - $rTTL_{TE}$(monitor), where $rTTL_{TE}$(monitor) represents the IP TTL of the ICMP `time exceeded` message received by the monitor.

On the contrary, if the return ingress LER sets the IP TTL to 64, for example in the case of a JUNIPER device that generates an `echo reply` message, the return PH will select the IP TTL, as 64 is much smaller that 255, and the tunnels are rather short (see Figure 4.12 in Chapter 4). Consequently, the tunnel no longer behaves as if the the IP TTL was propagated by the ingress LER (see Figure 6.2(a)). Therefore, computing RTL for the forward egress LER gives:

$$\begin{aligned} RTL &= L_R^T - L_R^P \\ &= (L'^T_R + L''^T_R + 1) - (L'^P_R + L''^P_R + 1) \\ &= L''^T_R - L''^P_R \\ &= L''^T_R \end{aligned} \tag{6.4}$$

Indeed, as already stated previously, the BGP return paths of the ICMP `time exceeded` and `echo reply` messages are identical ($L'^T_R = L'^P_R$, see Equation 6.2). Moreover, because of the MIN operation, the tunnel length on the return path is not taken into account in the IP TTL of the received `echo reply` message ($L''^P_R = 0$), while it is in the TTL of the `time exceeded` message ($L''^T_R \neq 0$). Similarly to L_R^T, L_R^P can be easily computed as 64 - $rTTL_{ER}$(monitor), where $rTTL_{ER}$(monitor) represents the IP TTL of the ICMP `echo reply` message received by the monitor.

Figure 6.6 illustrates an example of an MPLS network on which RFPLA and RTL can be applied. Considering the JUNIPER forward egress LER with signature $< 255, 64 >$, the three path lengths can be computed based on the IP TTLs of the ICMP messages received by the monitor. Knowing L_F^T can be directly obtained by subtracting 1 to the position of the egress LER in the forward trace[13], it comes:

$$\begin{aligned} L_F^T &= 3 - 1 = 2 \\ L_R^T &= 255 - rTTL_{TE}(\text{monitor}) = 255 - 251 = 4 \\ L_R^P &= 64 - rTTL_{ER}(\text{monitor}) = 64 - 62 = 2 \end{aligned}$$

Then, based on these values, we can compute RFPLA and RTL:

$$RFPLA = L_R^T - L_F^T = 4 - 2 = 2$$
$$RTL = L_R^T - L_R^P = 4 - 2 = 2$$

The values of RFPLA and RTL correspond to the length of the return LSP, as expected. Note that in the ideal case where the forward and return tunnels are identical, the two techniques determine also the length of the forward LSP.

[13]The forward LSP is invisible, and its LSR does not appear in the trace.

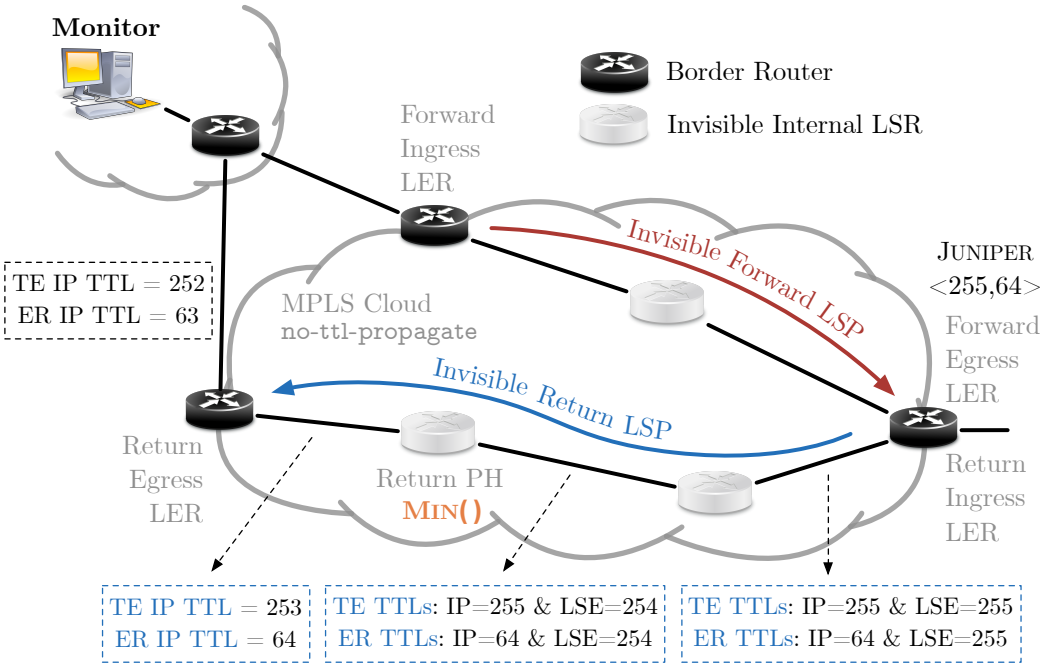


Figure 6.6: Evolution of the IP and LSE TTLs on the return path. The values are shown for the `time exceeded` (TE) and `echo reply` (ER) messages. The MIN operation at the return PH selects the LSE TTL for the ICMP `time exceeded` packet (the device decrements the LSE TTL before copying its value, as already mentioned in Section 6.2), and the IP TTL for the `echo reply` one.

By definition, RTL can only be computed for JUNIPER devices running Junos, and with $< 255, 64 >$ as signature. On its side, RFPLA is more generic, and applies thus to any configuration. However, as already mentioned, it is sensitive to BGP path asymmetry. Indeed, adding a new router between the monitor and the forward ingress LER in the example illustrated in Figure 6.6 is enough to modify its value. As a result, a deviation from 0 cannot be directly interpreted as the presence of invisible MPLS tunnels. Therefore, RFPLA should not be considered alone on a trace to avoid producing false positives (i.e., a tunnel length of X hops is inferred because the return path has X more hops than the forward one due to routing asymmetry) and false negatives (i.e., the return path is X hops shorter than the forward one). Instead, it should be computed on multiple traces collected by several independent monitors in order to infer the distribution of its values at the AS granularity. In this way, if `traceroute` enters through a sufficient number of border routers in the target AS, the impact of BGP should be mitigated. Indeed, in the absence of invisible MPLS tunnels in the AS, the distribution should look like a normal distribution centered at 0, as forward and return paths should have, on average, similar lengths. On the opposite, if the AS makes use the `no-ttl-propagate` option, the MIN(IP TTL, LSE TTL) operation causes return paths to be longer than the forward ones, and a significant shift should be observed towards positive values. Note that this shift provides the average LSP length in the AS.

As RTL gives the exact length of the return LSP, and does not depend on BGP configurations, it should be considered in priority compared to RFPLA, when possible.

6.4.2 Revealing the Hidden Hops

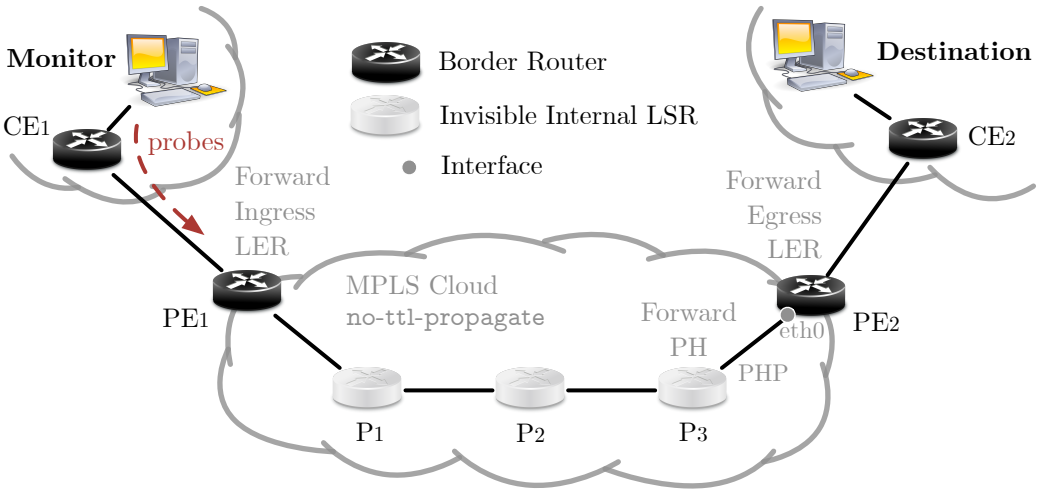
As explained in Section 4.2, network operators may deploy MPLS for transit traffic only. In this case, tunnels are only built towards the loopback addresses of the border routers, and packets sent to internal destinations are not forwarded with MPLS. This behavior is implemented by default in JUNIPER routers. Some network administrators may also decide to build an LSP for each prefix in the IGP routing tables (loopback addresses included), as done by default by CISCO. In this architecture, the traffic towards a destination inside the AS is also forwarded with MPLS.

The first revelation method, DPR[14], allows to expose the content of invisible LSPs in networks deploying MPLS for transit traffic only. Tunnels with a JUNIPER router as egress LER are thus perfect candidates for this technique. Note that in practice, CISCO devices can be configured to mimmic the behavior of JUNIPER equipment using the command `mpls ldp label allocate global host-routes`[15]. The principle of DPR is simple. Indeed, running `traceroute` towards the internal interface of the egress LER is enough to expose the content of the LSP, as in this kind of architecture, the traffic directed to a destination inside the domain is not forwarded with MPLS. It is illustrated in Figure 6.7. When `traceroute` is run towards the external destination, the tunnel is not visible in the output. However, when the internal interface of the egress LER is selected as the target, the different LSRs (P_1, P_2, and P_3) behave as standard IP routers, and send back ICMP `time exceeded` messages to the monitor. They are thus visible in the `traceroute` output. Note that, in the MPLS domain, the IP path followed by the probes sent towards the internal interface of the egress LER is identical to the LSP, as LDP uses the shortest IGP route towards the announced loopback address. However, both paths must start at the same border router.

The second revelation technique, BRPR, allows to reveal hidden LSPs in networks enabling MPLS for all the IGP prefixes (the standard and per default behavior of CISCO LSRs). Since `traceroute` naturally reveals the incoming IP interface of each egress LER, a recursive approach can be applied. It consists in targeting this last internal address to reveal each intermediate hop, in a backward fashion, until reaching the ingress LER. This approach is illustrated in Figure 6.7. The first `traceroute`, towards the external destination, identifies only the LERs of the tunnel. Then, when running again `traceroute` towards the internal interface of the egress LER, the network establishes a new invisible MPLS tunnel, with PE_1 and P_3 as ingress and egress LERs respectively. As P_3 is now the exit point of the tunnel, it replies to the probes, and appears in the

[14] DPR and BRPR were validated with GNS3 simulations (see Section 7.3.5) and cross-validated based on explicit MPLS tunnels (see Section 6.4.3).

[15] This command forces the router to only use loopback addresses with LDP. Note that, in order to allow transit traffic to be forwarded with MPLS based on loopback addresses, the BGP command `next-hop-self` should be used at the border routers (for both CISCO and JUNIPER devices).



Traceroute output (external destination):

To Destination:
1. CE1
2. PE1
3. PE2.eth0
4. CE2
5. Destination

Traceroute output (DPR):

To PE2.eth0:
1. CE1
2. PE1
3. P1
4. P2
5. P3
6. PE2.eth0

Traceroute outputs (BRPR):

To PE2.eth0:	To P3:	To P2:	To P1:
1. CE1	1. CE1	1. CE1	1. CE1
2. PE1	2. PE1	2. PE1	2. PE1
3. P3	3. P2	3. P1	3. P1
4. PE2.eth0	4. P3	4. P2	

Figure 6.7: Illustration of the revelation of a hidden LSP with DPR and BRPR. The return MPLS tunnel is not represented for clarity, as it does not influence the techniques. The three internal LSRs (P_1, P_2, and P_3) are hidden when targeting a destination outside the MPLS cloud. While DPR is able to reveal the whole LSP with a single `traceroute` to the egress LER (PE_2), BRPR needs several iterations.

output. A new `traceroute` can then be run towards P_3 in order to reveal P_2. BRPR continues performing these operations until no more LSR is revealed, as illustrated in the trace towards P_1. BRPR works well when the BGP routes remain similar for all internal prefixes of the targeted AS, i.e., when the packets enter via the same ingress LER and follow the same shortest IGP path inside the network (default LDP behavior). It is worth mentioning that the technique works because each intermediate target belongs to a prefix (the prefix of the link between the PH and the egress LER) also advertised by the PH of the current tunnel with LDP.

6.4.3 Validating the Measurement Techniques

The four techniques described in the previous sections were assessed through emulation testbeds with GNS3, a software allowing to run actual router OSs in virtualized routers [70]. This validation will be discussed in Section 7.3.5, as it takes into account different concepts and techniques that will be addressed in the next chapter.

To further validate DPR and BRPR, the two revelation techniques were cross-validated based on explicit MPLS tunnels. The data was collected on PlanetLab [32] using 23 monitors spread into five teams. Each team was responsible to run ICMP `paris-traceroute` towards 10,000 destinations (no overlapping between the teams) obtained from CAIDA's Archipelago dataset. A total of 269,096 `traceroutes` were performed, from which 14,771 distinct ingress-egress LERs pairs belonging to explicit MPLS tunnels were extracted. Note that both LERs of each pair had to be in the same AS, and that the LSP of each tunnel had to be fully revealed (i.e., no anonymous hops).

The two revelation techniques were applied on the collected pairs. On the one hand, DPR was considered as successful for a given tunnel if two conditions were met:

(i) The sequence of IP addresses obtained with the method between the ingress and egress LERs had the same length as the explicit LSP appearing in the traces collected during the measurement campaign[16].

(ii) This sequence did not contain any MPLS label.

On the other hand, BRPR was considered as successful if, at each step of the recursion, the last hop did not exhibit any label.

Note that BRPR and DPR could not be run for 9,407 out of the 14,771 distinct pairs, because either the ingress or the egress LER could not be retrieved. Table 6.2 summarizes the cross-validation on the 5,364 remaining pairs. We can observe that DPR and BRPR succeeded in 60% of the cases, but failed 8% of the time. For a few particular ingress-egress LERs (5%), tunnels were revealed partially by DPR, and partially by BRPR. In 26% of the cases, the LSP had only one

[16] In practice, most of the time, both paths (i.e., the explicit tunnel and the one revealed by DPR) were exactly the same. However, in some cases, similar paths with distinct IP addresses were observed if load balancing was performed based on ECMP.

Result	Proportion
BRPR or DPR failed	8%
DPR successful	57%
BRPR successful	3%
hybrid DPR/BRPR	5%
1Hop-LSP	26%

Table 6.2: Cross-validation results on 5,364 ingress-egress LER pairs scattered in 271 different ASs. The line "1Hop-LSP" represents the pairs with an LSP composed of single internal LSR, for which DPR and BRPR could not be distinguished.

internal LSR[17] (line "1Hop-LSP"). The revelation succeeded, but it was impossible to discriminate between DPR and BRPR. Finally, the table shows a very low success rate for BRPR (3%). This result suggests that, given the large proportion of CISCO routers deployed by operators, they are configured to only inject loopback addresses instead of all the IGP prefixes into LDP. It is particularly true when the network contains a mix of hardware (JUNIPER and CISCO). In that case, JUNIPER devices systematically filter any piece of information not associated to loopback addresses. Note that, in the survey, even if a large proportion of operators (i.e., 25%) use a mix of router technologies, CISCO devices are the most prominent (58%), followed by JUNIPER (28%).

6.5 Data Collection

When operators decide to hide the internal structure of their MPLS network, ingress LERs may seem to be directly connected to all egress devices. The number of neighbors (i.e. the degree) of border routers may thus appear as being artificially high. Consequently, the presence of *High Degree Nodes (HDNs)* in an AS may indicate the potential existence of an invisible MPLS cloud. Therefore, targeting such networks may avoid running the tunnel revelation methods on domains that do not deploy MPLS, and limits the probing cost of a `traceroute` campaign.

Prior to the deployment of the measurement techniques, HDNs were identified in CAIDA's ITDK dataset [29] collected in March 2016. The data was first cleaned by removing non publicly-routable IP addresses and pseudo-addresses allocated to non-responsive routers. After this pruning, 45,021,817 IP addresses, 44,700,863 nodes, 2,705,780 links, and 43,178 ASs remained.

In order to be considered as an HDN, a node had to to connected to a minimum of 128 neighbors. This threshold was selected because it was considered as a lower bound relative to well-known physical hardware, in particular PE routers which are expected to terminate invisible tunnels. For instance, the ASR9000 series is one of the best selling CISCO PE routers. This device can be equipped with up to 20 line cards, each containing up to 16 interfaces. Thus, a threshold of 128 was a reasonable balance between the volume of probes sent (no overloading of

[17] This observation is aligned with the distribution in Figure 4.12.

the network) and the amount of interesting data collected. Obviously, invisible tunnels can be established between nodes with a low degree, but many HDN pairs are expected to hide invisible tunnels (proportionally, much more than non-HDN pairs). With this threshold, 17,944 HDNs were identified in CAIDA's dataset.

To efficiently guide the measurements, the neighbors of the HDNs (set A – 599,467 unique nodes) and their own neighbors (i.e. the neighbors of the neighbors of the HDNs, set B – 983,793 unique nodes) were retrieved from CAIDA's ITDK dataset. This latter set allowed to provide IP addresses that did not belong to the same AS as the ones in the first set. Indeed, MPLS LERs should be located at the borders of domains[18]. Thus, in order to simulate transit traffic, meaning traffic entirely traversing the suspicious AS made of many HDNs, and ending in one of its neighbors, the different monitors had to target around HDNs. Consequently, the destinations of the measurement campaign consisted in the union of both sets (set $A \bigcup B$), which represented a total of 1,306,545 IP addresses.

The measurement scripts used `scamper` [96], and its `paris-traceroute` implementation with ICMP `echo request` packets, starting with a TTL equal to 2. In addition, `echo reply` probes were also sent to all IP addresses appearing in the traces for router signature inference[19]. For each of the collected traces, the last three hops, say `X`, `Y`, `D` (where `D` $\in A \bigcup B$) were extracted. `X` and `Y` were candidate endpoints of an invisible tunnel. Then, `paris-traceroute` was run again towards `Y`. If the corresponding trace ended with `X`, `H`, `Y`, the hop `H` was considered as revealed from an invisible tunnel, and was used as the new target of another `paris-traceroute` in an attempt to reveal more LSRs (recursive processing with BRPR). If a new hop, say `H'`, was discovered in the trace towards `H` (this trace ended with `X`, `H'`, `H`), the recursion continued with `H'` as the target, and so on. This process stopped either if no new address was revealed, or if a trace did not cross `X`. Note that multiple IP addresses could have been revealed in a single shot, i.e., the trace towards `Y` ended with `X`, $\mathtt{H}_1$, $\mathtt{H}_2$, ..., $\mathtt{H}_{n-1}$, $\mathtt{H}_n$, `Y`, with $n > 1$ being the number of discovered hops (DPR). It is worth mentioning that, as already said previously, it was impossible to distinguish DPR from BRPR if the recursion stopped at the second trace, as the revealed LSP had only a single hop.

Since the aim was to reveal invisible MPLS tunnels spanning a single AS, with HDNs as triggers, the post-processing methods selected only the traces ending in `I`, `E`, `D` where `I` and `E` were HDNs located in the same AS. The AS resolution was performed based on CAIDA's node-to-AS mapping. If an address could not be mapped by CAIDA, it was submitted to the IP-to-AS mapping tool of Team Cymru [140].

The tool was deployed on 91 PlanetLab monitors (also named *Vantage Points (VPs)*) scattered all around the world (USA, Canada, Europe, Japan, Russia, Brazil, China, Australia, and New Zealand). These monitors were equally distributed in five groups, paying attention to their geographical locations. Each groups was assigned a subset of destinations as follows:

[18] This assumption was verified with CAIDA's bdrmap [97] dataset, but partially. Indeed, both measurement campaigns were strongly different, and the intersection between the datasets was sometimes weak.

[19] Fingerprinting in computer networks was described in detail in Chapter 3.

(i) The neighbors of the HDNs (set A) were randomly spread over five subsets.

(ii) The neighbors of each neighbor (set B) were added in the corresponding subset.

The five destination subsets were thus consistent, i.e., if neighbor N is in the first set, then all neighbors of N are also in the first set. Their sizes were also similar: 579,012 (set 1), 583,173 (set 2), 586,363 (set 3), 588,771 (set 4), and 586,229 (set 5). All measurements were launched simultaneously from all monitors on November 18$^{\text{th}}$, 2016 with `scamper` probing at a rate of 25 packets/second. The fastest set of monitors sent its last probes on November 29$^{\text{th}}$, 2016, while the slowest finished on December 6$^{\text{th}}$, 2016.

6.6 Measurement Results

This section provides three kinds of analysis to demonstrate the effectiveness and accuracy of the measurement techniques. First, we will compare the performance of the two most powerful methods, DPR and BRPR, that are able to reveal the content of hidden LSPs (Section 6.6.1). Then, we will use RFPLA to analyze the difference between return and forward paths in terms of number of hops (Section 6.6.2). We will also cross-validate the technique when it intersects with DPR and BRPR. Finally, we will study the length of the return tunnels with RTL (Section 6.6.3). Again, we will cross-validate the method using DPR and BRPR with a return-and-forward-path-asymmetry perspective. In short, this section will demonstrate that most of the invisible tunnels can be identified in some way, either explicitly (DPR or BRPR) or implicitly (RFPLA or RTL).

Table 6.3 provides many pieces of information for the ASs having the largest number of HDNs, and for which the content of invisible MPLS tunnels could be revealed (with the exception of AS2856). The two columns labeled "HDNs" refer to the number of high-degree nodes found in CAIDA's dataset ("ITDK"), but also to those ("Cand.") that were encountered in the measurement campaign and that could potentially act as ingress or egress LERs. The "I-E pairs" columns refer to pairs of IP addresses, belonging to candidate HDNs, that could have been ingress or egress LERs ("Cand"). The column "%Rev." provides the proportion of candidate ingress-egress pairs between which an invisible LSP could be revealed. The next three columns provide raw statistics about the LSPs discovered between the pairs. The column labeled "#LSPs" gives the number of unique LSPs (as sequences of IP addresses) that were revealed, while the column "#LSRs" gives the number of unique IP addresses belonging to the discovered LSRs. The last column "%LERs" is the proportion of those revealed IP addresses also identified as ingress or egress LERs. Finally, the column "Graph Density" indicates how the density[20] inference of the graphs of those ASs could be corrected based on the revelation of the invisible MPLS tunnels. It is worth noting that the density is computed here only based on ingress-egress pairs (and not on the whole ISP graph).

[20]The density of a graph with E edges and V vertices is $\frac{2\times E}{V\times(V-1)}$.

ISP (ASN)	HDNs		I-E Pairs (IP)		Revealed LSPs (IP)			Graph Density	
	ITDK	Cand.	Cand.	%Rev.	#LSPs	#LSRs	%LERs	Before	After
Telia (1299)	1,819	1,317	58,548	0.2	102	59	42.4	0.024	0.019
China Tel. (4134)	1,212	1,078	31,728	2.8	1,016	281	61.6	0.008	0.007
Tinet Spa (3257)	1,032	654	12,411	55.1	12,577	1,092	44.2	0.033	0.009
Level 3 (3549)	708	425	9,028	65.6	8,675	757	32.6	0.065	0.007
Deutsche Tel. (3320)	497	364	21,189	68.2	29,395	1,385	40.0	0.108	0.013
Telecom Italia (6762)	346	129	6,235	73.6	7,548	214	83.6	0.236	0.094
Qwest (209)	271	110	1,609	28.0	552	65	0.0	0.151	0.056
Bharti Airtel (9498)	159	150	11,909	12.5	4,199	493	44.8	0.138	0.041
PCCW Global (3491)	92	57	3,512	52.6	3,704	264	5.3	0.300	0.045
British Tel. (2856)	1,944	148	5,656	0.1	3	0	0.0	0.200	0.200

Table 6.3: Invisible MPLS tunnels discovery for different ASs of interest. Note that "I-E" stands for ingress-egress, while "Rev." and "Cand." mean respectively revelation and candidate. Most ASs are Tier-1 or Transit (Tier-2, etc.) ISPs having numerous interconnections, possibly resulting in dense HDN graphs.

6.6.1 Path Revelation with DPR and BRPR

During the measurement campaign, a total of 13,771 invisible tunnels were revealed. More specifically, 8,477 of them were discovered by DPR, 2,270 by BRPR, while the remaining 3,024 were too short to determine which technique applied, as their LSP was composed of only one internal LSR. Note that the additional measurement cost induced by BRPR (i.e., the recursion to reveal the tunnel content) was 8,180 probes.

Figure 6.8 illustrates the length of the revealed tunnels, i.e., the number of internal LSRs (X-axis) in their LSP. The Y-axis provides the raw number of IP addresses acting as egress LERs. The red dot refers to very short tunnels having only a single LSR. In this situation, DPR and BRPR are indistinguishable. The distribution does not really look like a power law with a strong shape and heavy tail. However, it is still a strongly decreasing function bounded with relatively short tunnels. Indeed, very few LSPs have more than 12 LSRs. This tunnel length distribution is aligned with previous results on visible tunnels (see Figure 4.12 in Chapter 4). We can also observe that the curves of DPR and BRPR behave differently. The reason is that DPR discovers the whole tunnel with only one additional `traceroute`, while BRPR needs one for each IP address. A significant number of its attempts may fail before discovering the whole tunnel, resulting in shorter tunnels on average. Table 6.3 shows the number of IP addresses belonging to the LSRs revealed during the campaign (column "#LSRs").

Finally, Figure 6.9 shows the RTT evolution for each hop of a trace crossing an invisible MPLS tunnel in AS3549. When the tunnel is invisible (blue line), a jump of about 50 ms can be observed between hops 8 (ingress LER) and 9 (egress LER). However, once the LSP has been revealed (black curve), this large delay is actually decomposed between the seven hops of the tunnel.

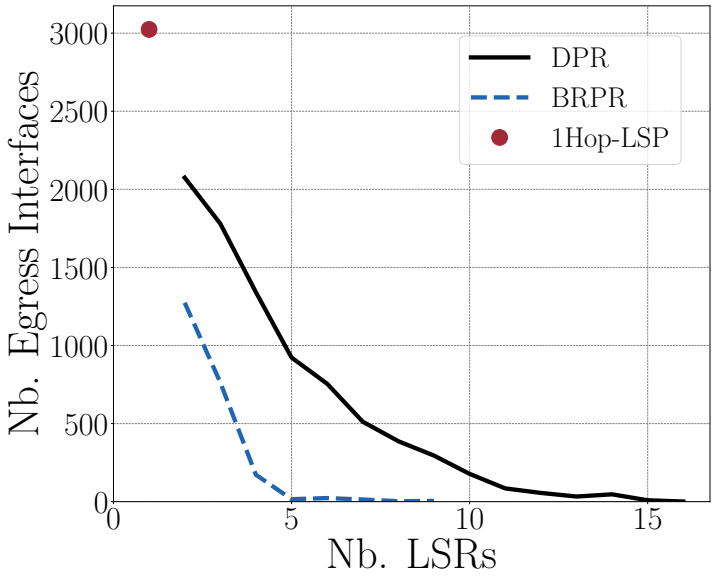


Figure 6.8: Forward tunnel length distribution.

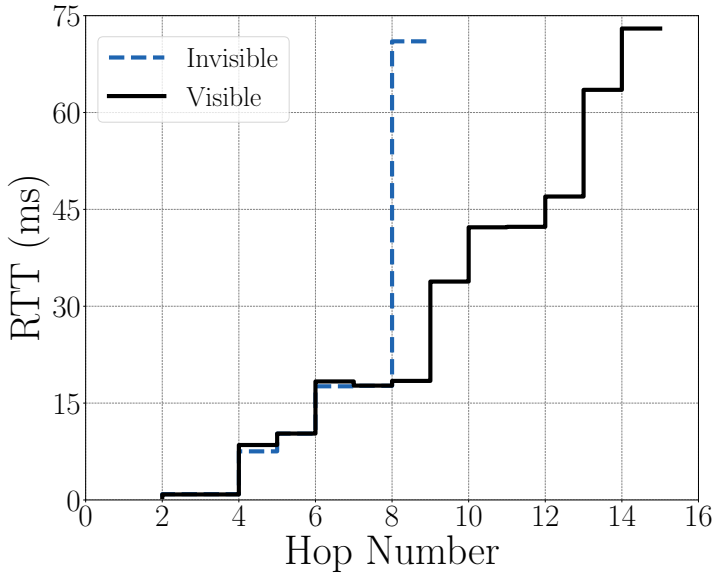


Figure 6.9: RTT correction thanks to a tunnel revelation (AS3549).

6.6.2 Return vs. Forward Asymmetry

As already mentioned in Section 6.4.1, RFPLA is sensitive to BGP route asymmetry, and should therefore be studied at a large scale (e.g., on a whole AS). Figure 6.10 provides the distribution of RFPLA values for the dataset collected during the measurement campaign. As a reminder, a value of 0 is inconclusive, as it means that return and forward paths have the same length, and therefore, do not contain any invisible tunnel. Similarly, a negative value implies that the return path is shorter than the forward one, and we can not conclude that the network hides MPLS routers. Finally, a positive value is the ideal case, as the return path is longer than the forward one. The presence of an invisible MPLS cloud can be assumed.

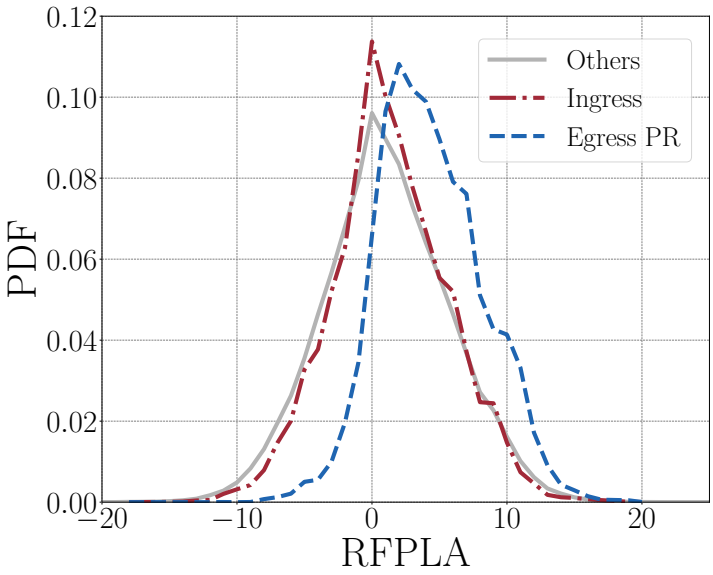


Figure 6.10: Distribution of RFPLA values for the measurement campaign, where “PR” means *Path Revelation*

The figure provides the distribution in several cases. The grey curve (“Others”) represents the values computed for the IP addresses that do not belong to HDNs, while the red one (“Ingress”) is associated to the addresses of routers identified as potential ingress LERs[21]. In both situations, the path asymmetry follows a normal law centered at 0, with a median value of 1, as the symmetry is not perfect. Indeed, in general, paths between two nodes in the Internet are not the same in both directions. It is due to, among others, BGP hot-potato routing. However, over a large number of pairs, the distribution should be, on average, (almost) symmetrical.

The conclusions are different in the case of IP addresses belonging to the egress LERs of invisible MPLS tunnels revealed during the campaign (blue curve, “Egress PR”, in the figure). Indeed, the normal law is now significantly shifted towards positive values (median of 4). This shift is a direct consequence of the forward tunnel invisibility. Indeed, the forward path length

[21]Nodes identified as potential ingress or egress LERs are HDNs.

does not include the hidden hops, while the complete return path length is taken into account in the TTL of the ICMP replies sent by the egress LER. The addresses of potential egress LERs that did not lead to any tunnel revelation were also considered. The corresponding distribution is shown in Figure 6.11 (grey curve, "Egress NPR"). In this case, the shift towards positive values is much more limited. Consequently, RFPLA (and, indirectly, RTL) seems not really efficient when path revelation does not work either.

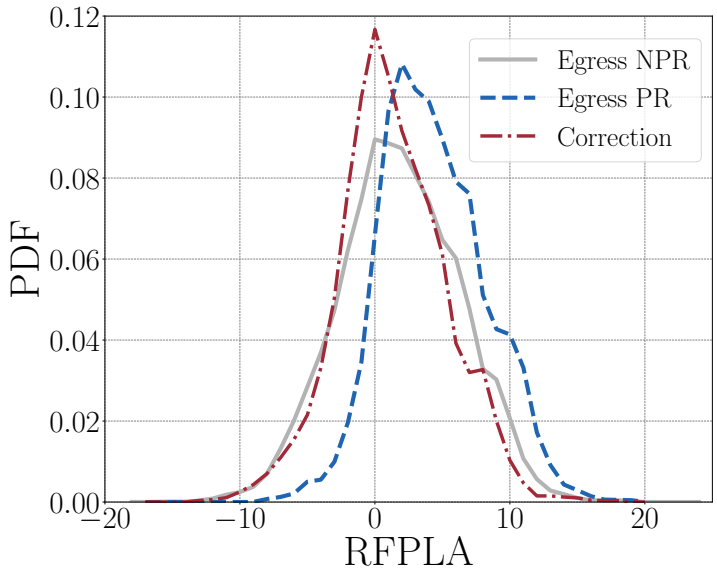


Figure 6.11: Corrected distribution of RFPLA values for the measurement campaign, where "PR" and "NPR" mean respectively *Path Revelation* and *No Path Revelation*.

Figure 6.11 tries to fix the shift observed for real egress LERs using the actual lengths of the forward paths revealed by DPR or BRPR. This cross validation was performed on the intersection of the revelation methods and RFPLA. For each revealed LSP towards an egress LER, the number of discovered LSRs was added to the forward path length, and RFPLA was reassessed. This correction allows to show that RFPLA works very well for most networks. In particular, the corrected distribution (red curve, "Correction") is almost centered at 0 compared to the two other curves. Again, the shift in asymmetry is much more noticeable when looking only at forward LSPs discovered with a path revelation mechanism (blue curve, "Egress PR"). It confirms that RFPLA is much more relevant for egress LERs that are part of a revealed tunnel, than for the ones that aren't. These results tend to show that both kinds of technique (revelation and length analysis) apply to the same network configurations.

6.6.3 Return Tunnel Length

On its side, RTL is more accurate than RFPLA, as it is not sensitive to BGP asymmetry. However, it can only be applied on LERs with $< 255, 64 >$ as signature (instead of all LERs for RFPLA). It

allows to determine the exact return tunnel length using the fact that the hidden LSRs are taken into account in the IP TTL of the ICMP `time exceeded` message, while they aren't in the TTL of `echo reply` one (see Section 6.4.1).

Figure 6.12 shows the asymmetry between the return and forward paths of `traceroute` and `ping` messages. Only JUNIPER egress LERs related to revealed invisible tunnels were considered. The comparison of the two curves shows a shift towards the positive values for the distribution related to `traceroute` (ICMP `echo request` and `time exceeded` messages). Indeed, the path asymmetry does not follow a normal law centered at 0 (blue curve, "`traceroute`"), as the median equals 4. The reason is that the MPLS tunnel on the forward path remains unnoticed, while the length of the return tunnel is taken into account in the IP TTL of the `time exceeded` message (the MIN operation selects the LSE TTL, as the IP TTL remains at 255 until the end of the tunnel). On the contrary, the distribution is almost centered at 0 in the case of `ping` (ICMP `echo request` and `echo reply` messages). Its highest peak is at 0 while the median equals 2 (black curve, "`ping`"). Indeed, in this case, the return tunnel length is not reflected by the IP TTL of the `echo reply` message received by the monitor (the MIN operation selects the IP TTL, as 64 is much smaller than the value of the LSE TTL).

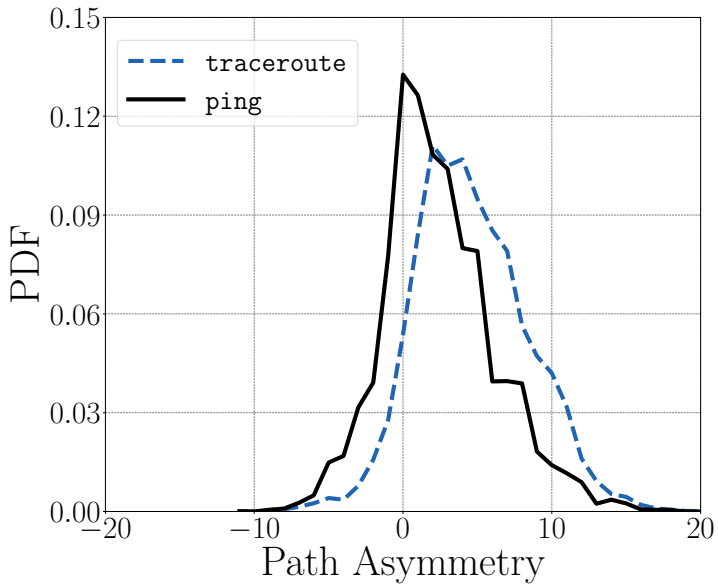


Figure 6.12: Path asymmetry for `traceroute` and `ping` messages. Only JUNIPER egress LERs associated to revealed invisible tunnels are considered.

Figure 6.13 shows the distribution of RTL values for the IP addresses of JUNIPER egress LERs belonging to tunnels revealed during the campaign. By definition, it reflects the actual length of the invisible return LSPs. Therefore, it can be compared with the length of forward LSPs given in Figure 6.8. Both distributions look very similar. Note that the low amount of negative values (the shaded area) in Figure 6.13 comes probably from ECMP variations, or from

other path noise specific to some monitors.

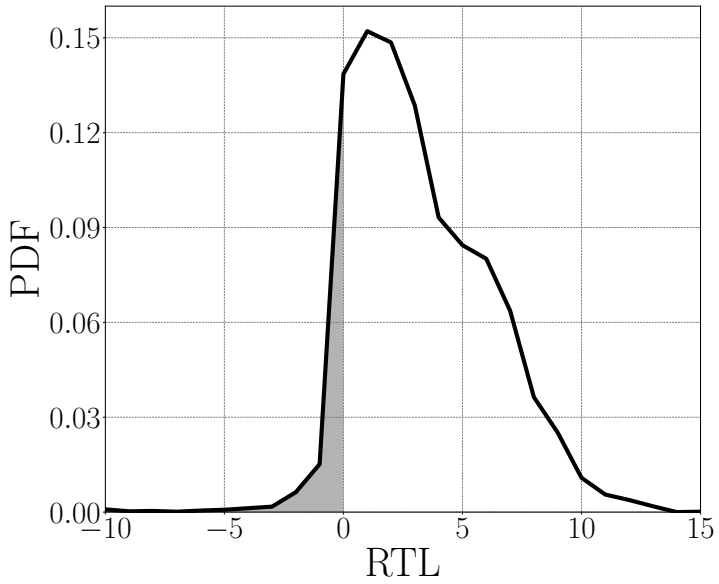


Figure 6.13: Distribution of RTL values for the measurement campaign.

Finally, Figure 6.14 try to assess the accuracy of RTL. In the same fashion as for RFPLA, the actual forward tunnel length, obtained either with DPR or BRPR, was subtracted to the return tunnel length. As the distribution almost follows a normal law centered at 0, the length of the return LSPs inferred by RTL is similar to the one of the forward LSPs. Consequently, RTL seems to work well at a global scale.

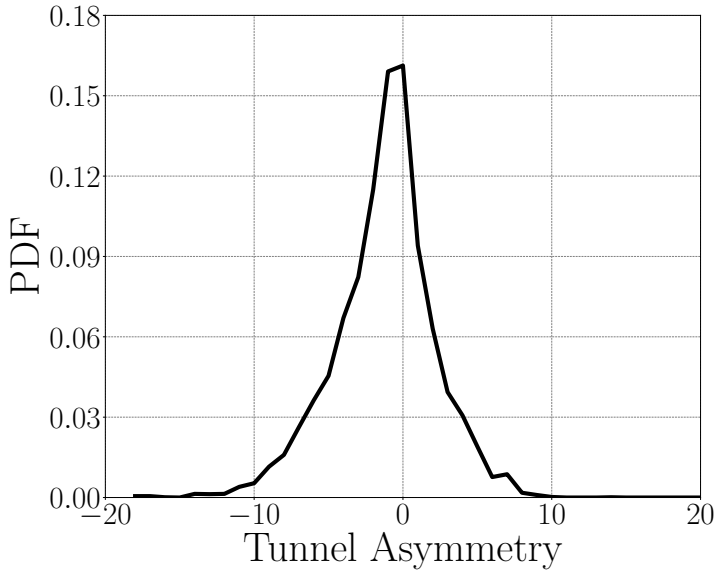


Figure 6.14: Tunnel asymmetry based on RTL values for the measurement campaign.

6.7 MPLS Analysis

The deployment of MPLS is greatly variable from one ISP to another, as shown in Table 6.4. This table focuses on the same networks as the ones presented in Table 6.3. The entries are sorted according to the TTL signatures of their network equipment (see Table 3.1).

ASN	TTL signature (%)			LSR Revelation (%)				#Rev. LSRs (median)		
	<255,255>	<255,64>	<64,64>	DPR	BRPR	1Hop-LSP	Multi	RFPLA	RTL	FTL
3491	93	0	0	2	74	20	4	4	-	2
4134	73	0	0	13	3	83	1	1	-	1
2856	67	30	1	33	0	67	0	−3	-	1
3320	53	41	0	50	9	2	40	4	2	2
6762	37	53	0	6	69	17	7	4	3	2
209	27	37	0	98	0	2	0	3	2	4
1299	25	74	0	19	3	77	0	0	0	1
3549	11	45	38	73	3	1	24	5	4	5
9498	7	72	0	99	0	1	0	4	4	4
3257	0	96	0	99	0	1	0	4	2	4

Table 6.4: MPLS deployment per AS, in terms of IP addresses, where “TTL signature” represents the TTL signature distribution for the IP addresses in each AS, “LSR Revelation” gives the proportion of hidden IP addresses discovered by each revelation method, and “#Rev. LSRs” shows the median value of the return LSP length estimated by RFPLA and RTL, as well as the median length of the forward tunnels revealed by DPR and BRPR (“FTL”). Note that the percentage for TTL signatures is rounded (the total may exceed 100%), “1Hop-LSP” refers to hops revealed by either DPR or BRPR (impossible to discriminate between the two techniques for LSPs with a single LSR), and “Multi” refers to LSRs revealed by different discovery techniques (in some cases, by DPR, and in others, by BRPR).

The first observation is that first ranked ASs deploy mostly CISCO routers, while last ranked ones use mainly JUNIPER devices. Several of them show a consistent behavior. For instance, AS3257, and to a lower extent AS9498, are built around JUNIPER hardware. As expected, most of their hidden IP interfaces are revealed with DPR. On the contrary, AS3491 deploys mostly CISCO hardware, and its hidden LSRs are discovered by BRPR most of the time. Other ASs appear to deploy a mix of network equipment. As already mentioned in Section 6.4.3, DPR provides better results for them. It is worth noting that AS3549 is the only one with a high proportion of devices with signature <64,64>. The most efficient revelation method for this AS is DPR. Consequently, the behavior associated to this signature looks similar to the one of JUNIPER routers. Another finding for AS3549 (not shown in the table) is that JUNIPER devices seem mostly deployed at the edges of the network (ingress and egress LERs), while the equipment with signature <64,64> is essentially present in the core (revealed IP addresses).

The comparison of the forward and return tunnel lengths, given by the last three columns of the table, is also interesting. Note that AS2856 is not significant for this analysis, as almost no

tunnel was revealed for this AS (see Table 6.3). The median return tunnel length estimated by RFPLA is not far from the median length of the revealed forward LSPs, considering that this method is sensitive to asymmetric routing. On its side, RTL provides a value which is consistent with the forward tunnel length. These results confirm that both techniques are able to signal the presence of invisible MPLS clouds. Finally, AS1299 seems to mainly contain really short tunnels (77% of the LSRs are classified as "1Hop-LSP", meaning that only one LSR was retrieved). It may explain, at least partially, why RFPLA and RTL do not provide significant information for this AS.

6.8 Internet Model Update

The degree of a node, i.e. its number of neighbors, is an important metric for Internet modeling. When computed over a whole IP domain, its distribution may be an indicator of the network resilience to failures and attacks [74]. Faloutsos et al. [61] were the first to highlight the power-law shape of this distribution over the Internet. This result has been heavily questioned in multiple studies [39, 93, 107] in the past. In this chapter, we made the assumption that invisible MPLS clouds may influence the node degree distribution. Indeed, since each ingress LER may artificially appear as directly connected to all egress LERs, the network may turn into a dense mesh of high-degree nodes. This assumption was the starting point of the measurement techniques for revealing hidden LSRs.

Figure 6.15 shows the effect of hidden tunnels on the degree distribution, and how this distribution is corrected once their content is taken into account. In order to perform this analysis, the ingress and egress LERs of each revealed MPLS tunnel was mapped to a router identifier using CAIDA's ITDK dataset. A first graph was built based on this information (the revealed LSPs were not yet taken into account), and was used to compute the degree distribution in the invisible case (blue dashed line in the figure). In a second step, all the IP addresses discovered by DPR and BRPR were also mapped to a router identifier. They allowed to create an updated version of the first graph taking into account the content of the tunnels. It was used to compute the degree distribution in the visible case (black line in the figure). During the process, if an IP address could not be mapped to a router identifier based on CAIDA's dataset, it was assigned a new value. However, 97% of the revealed IP addresses could be found in CAIDA's dataset. In order to obtain the degree distribution for both graphs, the number of neighbors was computed for each node. Note that only the intersection between the dataset collected with PlanetLab and the ITDK one was considered. It justifies why the degrees observed in Figure 6.15 are much lower than the ones in Figure 6.1. Indeed, the PlanetLab dataset is limited to a relatively small sample. On Figure 6.15, the Y-axis provides the PDF, while the X-axis gives the number of neighbors.

Figure 6.15 illustrates the degree distribution for all the ASs observed during the campaign. Two main results arise. First, as expected, the proportion of HDNs decreases once the content of

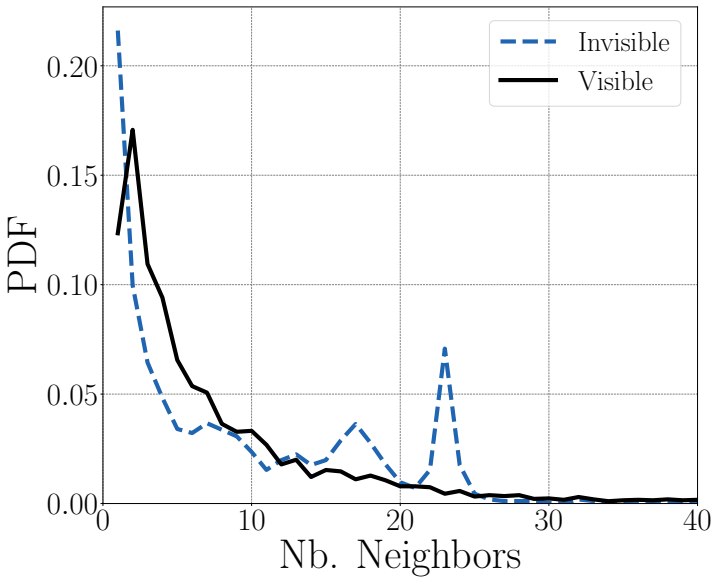
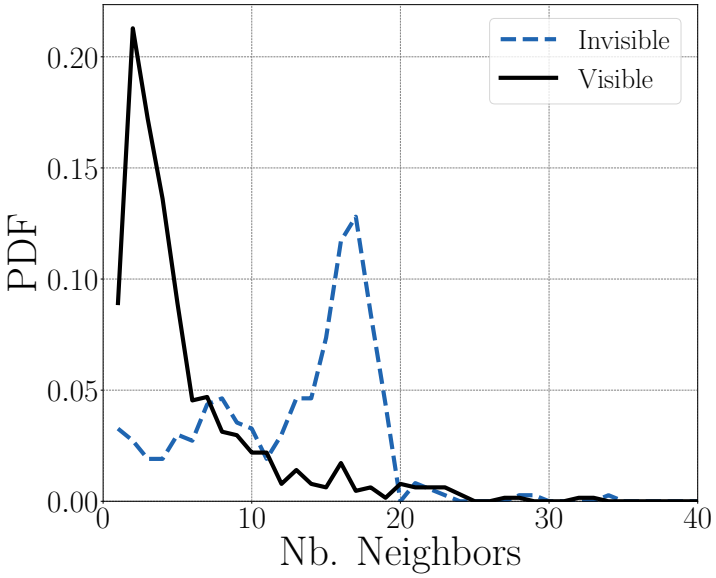


Figure 6.15: Effect of invisible MPLS tunnels on the node degree distribution.

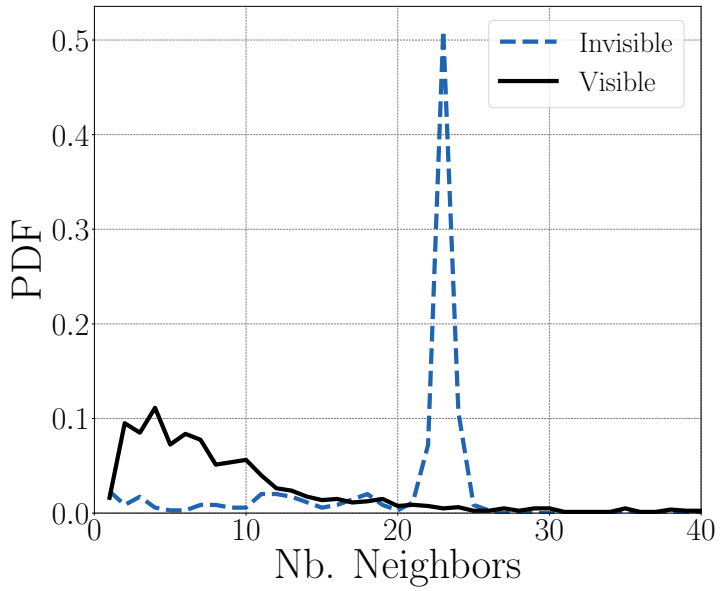
the hidden tunnels is taken into account. Then, two peaks are observed for 17 and 23 neighbors. They are due to two ASs in particular. The peak at 17 is caused by invisible tunnels in AS3549 (LEVEL3), while the other is due to AS3320 (DEUTSCHE TELEKOM), as proved by Figure 6.16. Focusing on a given AS provides very insightful results. Indeed, a deeper analysis of AS3320 revealed a kind of full-mesh made of 23 routers (a representative sample of the real network). This mesh could be reduced thanks to the discovery techniques so that the degree distribution became standard. This result is confirmed by the graph density analysis provided in Table 6.3. Indeed, its density is divided by a factor of ten once invisible tunnels are revealed.

The path length is also an important metric for modeling the Internet topology. Indeed, it allows, for example, to determine the path offering the minimum distance between a given pair of devices (i.e., the shortest path), or the diameter of a network (i.e., the longest of all shortest paths in the network). Obviously, if many nodes are hidden by long invisible MPLS tunnels, the path length distribution and the resulting inferred model are biased.

Figure 6.17 shows the effect of invisible MPLS tunnels on the path length distribution computed over the PlanetLab dataset. The black plain line (“Visible”) takes into account the LSRs revealed by DPR and BRPR, while the blue dashed line (“Invisible”) doesn’t. If both distributions display more or less the familiar bell-shaped curve typical of Internet distance distributions, it is clear that, revealing hidden hops causes a shift towards longer routes. In particular, the mean is at 9 with invisible tunnels, while it is at 11 when when their content is taken into account. However, it is worth noting that the corrected distribution still underestimates the actual path lengths, since when a trace crosses several invisible MPLS tunnels, the techniques only reveal the last one. Thus, as a significant number of traces are likely to cross up to two



(a) AS3549 (LEVEL3).



(b) AS3320 (DEUTSCHE TELEKOM).

Figure 6.16: Effect of invisible MPLS tunnels on the node degree distribution for two specific ASs.

invisible MPLS networks, the actual length shift should be larger. Finally, as most of the targets in the measurement campaign were egress LERs belonging to Tier-1 networks, the obtained path lengths should be multiplied by almost a factor of two in order to get closer of the typical route length in the Internet. Indeed, the descending path from the Tier-1 network to another Stub network was not taken into account.

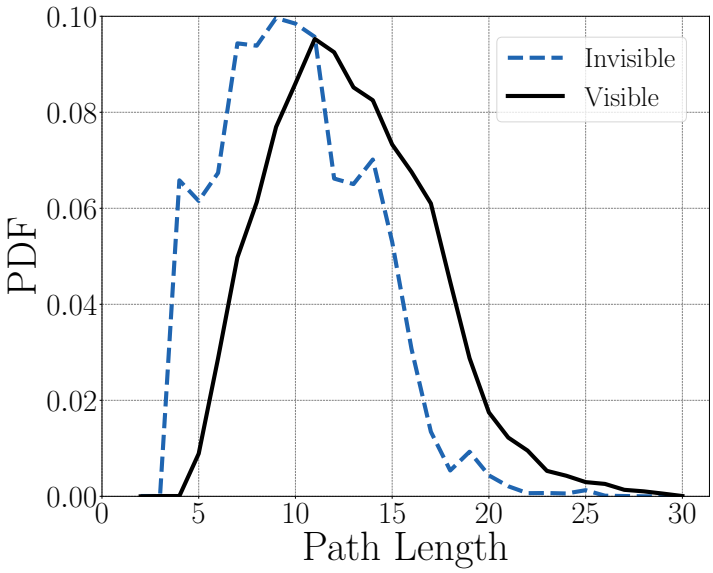


Figure 6.17: Effect of invisible MPLS tunnels on the path length distribution.

Obviously the results presented in this section are only illustrations of the effect of invisible tunnels on basic graph characteristics, as observed in the dataset. Much more extensive measurement campaigns and analyses are required to get more precise results at the Internet scale.

6.9 Conclusion

This chapter presented several kinds of techniques for inferring and revealing IP level information hidden by invisible MPLS tunnels. After a complete description of their functioning, we validated them based on a network operator survey, and through cross-validation[22]. We also analyzed a measurement dataset obtained thanks to their implementation and deployment on PlanetLab. The results allowed us to get more insight into standard MPLS practices of ISPs. In addition, we also revisited the inference of some basic Internet graph characteristics that are biased due to hidden LSRs.

The four measurement mechanisms can be classified into two categories. First, RFPLA and RTL are able to estimate the length of invisible tunnels on the return paths. On the one hand,

[22]Their validation through GNS3 emulations will be described in Section 7.3.5.

RFPLA has the advantage of being scalable. Indeed, it is a pure analytical technique able to work with any IP level dataset, as it only relies on standard `traceroute` data. On the other end, RTL requires an additional `echo request` per IP address, and can only be applied on JUNIPER devices with signature $< 255, 64 >$. However, when applicable, it provides a more accurate estimation of the return tunnel length than RFPLA, which is sensitive to BGP path asymmetry. These two mechanisms are suitable to determine whether an AS hides an invisible MPLS cloud, or to evaluate the impact of invisible MPLS tunnels on the length of Internet paths.

On their side, DPR and BRPR reveal the LSRs of hidden LSPs on the forward paths. They require a more specific and complex measurement campaign, since additional `traceroutes` must be dynamically run in order to perform the revelations. These techniques allow to gain knowledge on the internal architecture of invisible MPLS clouds. In addition, the Internet graph, and its node degree distribution in particular, can be corrected.

Note that the measurements mechanisms could not identify any invisible tunnel in a few ASs claiming to deploy MPLS (according to their websites). The most plausible reason is that they probably use MPLS with UHP only, for VPN and/or traffic engineering, and their hidden LSPs remained thus unnoticed.

Finally, the measurement campaign was driven by the suspicious high-degree nodes in the router-level topology. Indeed, the different attempts to reveal invisible tunnels were performed around HDNs. This solution is not scalable in practice, as it requires the router-level graph of the target network. However, we have seen that RFPLA and RTL are able to infer the presence of invisible MPLS clouds. Consequently, theses techniques could be used as triggers in a tool performing `traceroute` measurements. In this way, if RFPLA and RTL determine that LSRs may be hidden between a given pair of routers, the tool could run DPR and BRPR to try to reveal the content of the potential invisible MPLS tunnel, as suggested by Table 6.5. This topic will be addressed in the next chapter.

Brand	**MPLS (by default)**		**Trigger**		**Revelation**	
	LDP	Popping	RFPLA	RTL	DPR	BRPR
CISCO	all prefixes	PHP	✓	–	–	✓
JUNIPER	loopback	PHP	(✓)	✓	✓	(✓)

Table 6.5: Measurement techniques applicability.

CHAPTER 7

CLOSING THE LOOP WITH TNT: REVEALING ALL MPLS TUNNELS

The previous chapter presented techniques for exposing the content of MPLS tunnels hidden to `traceroute` in the particular context of *Penultimate Hop Popping (PHP)*. However, in practice, some network operators may decide to configure their egress LERs in order to perform *Ultimate Hop Popping (UHP)*. As a first step, this last chapter focuses on the detection and discovery of UHP invisible tunnels. Next, it introduces *Trace the Naughty Tunnels (TNT)* [143], an extension to `paris-traceroute` for revealing most (if not all) MPLS tunnels along a path. As such, `TNT` appears as an integrated tool able to identify explicit, implicit, opaque and invisible tunnels. It works in two basic stages. First, it runs `paris-traceroute` towards the specified destination, detects all MPLS routers that are directly identifiable in its output, and looks for potential evidences of the presence of hidden LSRs. Those evidences are suspicious patterns, such as abrupt and significant shiftings of the TTL values. Second, if the tool determines that a tunnel may be hidden on the path based on those evidences, it launches additional probes for possibly revealing its content. After a more complete description of `TNT`, we will discuss its validation through emulation with GNS3. Then, we will discover how it could be calibrated through a dedicated measurement campaign. Finally, we will analyze data collected by the tool on CAIDA's Archipelago platform, and provide a first quantification of MPLS tunnels, updating so the state-of-the-art view of MPLS deployment in the Internet.

7.1 Introduction

As already stated in the previous chapter, Internet topology discovery has attracted great attention from the research community [52, 76] during the last years. Numerous tools have been suggested to better capture the Internet architecture at the IP and router levels, mainly based

on `traceroute` and alias resolution techniques. The collected data is used to model the Internet [116], and allows to get a better knowledge of the network ecosystem and its organization.

However, in practice, the measurement process based on `traceroute` is not perfect. Several improvements have already been suggested by the research community. For instance, DOUBLETREE [20, 54] reduces the probing redundancy by allowing a cooperation between scattered monitors. On its side, `paris-traceroute` [10] fixes issues related to IP load balancing, avoiding so the inference of false links between IP interfaces. The presence of *middleboxes* [56] along a path can be revealed with `tracebox` [50], an extension to `traceroute`. YARRP [19] provides techniques for speeding up the probing process. Reverse `traceroute` [86] is able to determine the reverse path from the target back to the monitor. PASSENGER [136] and DISCARTE [135] extend `traceroute` with the IP Record Route option. Marchetta et al. [102] have suggested using ICMP Parameter Problem messages in addition to this Record Route option. They also exploited the ICMP Timestamp option in their tool DRAGO [103]. The identification of Layer-2 devices connecting routers has been addressed too, with a, nowadays, deprecated tool (i.e., IGMP-based probing) [107]. Finally, `tracenet` [59] mimics `traceroute` for discovering subnetworks.

The previous chapter introduced two TTL-based techniques (RFPLA and RTL) allowing to infer the presence of invisible tunnels performing PHP, as well as two revelation mechanisms (BRPR and DPR) able to expose their content. As stated in the conclusion, the inference techniques could be used as triggers in a measurement tool in order to determine if a tunnel revelation must be attempted between two hops in a trace. This chapter puts this idea into practice, and goes even further. It introduces `TNT` (*Trace the Naughty Tunnels*), an open-source extension to `paris-traceroute` that allows to identify any MPLS tunnel on the path between a source and a destination. More specifically, it includes techniques able to infer the presence of each type of tunnel presented in Chapter 2, as well as revelation mechanisms to expose the content of invisible ones, even in case of *Ultimate Hop Popping (UHP)*[1].

The contributions are fourfold:

(i) Improvement of the state of the art with `traceroute`-based **measurement techniques** able to reveal most (if not all) MPLS tunnels, even those that were built for hiding their content. Those techniques work with *indicators* or *triggers* that are used to determine the potential presence of a tunnel. When a trigger is pulled during a `traceroute` exploration, a *revelation* is attempted, with the objective of exposing the tunnel content. These indicators, triggers, and revelation techniques were validated using GNS3. We will also demonstrate that the different discovery mechanisms are efficient in terms of cost (i.e., the additional amount of probes injected in the network is reasonable, especially compared to the value of the discovered data) and errors (false positives and false negatives).

(ii) **Implementation** of those techniques within `scamper` [96] as a `paris-traceroute` exten-

[1] This mechanism was introduced in Section 1.5.5

sion called TNT, and deployment of this tool on CAIDA's Archipelago infrastructure [38]. TNT aims at replacing the old version of `traceroute` implemented within `scamper`, and is therefore subject to be run every day towards millions of destinations. Consequently, it will be useful for the research community in order to study the deployment and use of MPLS over time, and increase the knowledge on this technology.

(iii) **Analysis** of a dataset collected on Archipelago, and new quantification of the deployment of MPLS in the wild. The results of previous works [53] will therefore be corrected and updated.

(iv) **Reproducibility**. The source code of TNT, the GNS3 validation, the data processing and analysis scripts, as well as the collected dataset are publicly available [55].

In the remainder of this chapter, we will first update the classification presented in Section 2.3 in order to fix the model of opaque tunnels, and include the UHP behavior (Section 7.2). Then, TNT will be formally introduced (Section 7.3), and we will discuss its calibration based on the values of its parameters (Section 7.4). Finally, we will analyze the data collected by the tool on the Archipelago platform, and update the quantification of MPLS tunnels in today's Internet (Section 7.5).

7.2 MPLS Tunnels Classification Update

Chapter 2 introduced the measurement-based classification of MPLS tunnels suggested by Donnet et al. [53]. As already mentioned in Section 2.3, the models for opaque and invisible tunnels suffer from several limits.

By definition, routers in opaque tunnels implement the ICMP extension described in RFC4950. According to Donnet et al., this kind of tunnel is caused by operators that enable PHP and disable the TTL propagation. An element that goes against this hypothesis arose in Section 3.3.2. Indeed, the fingerprinting method showed that these specific tunnels appear with the use of CISCO equipment. This observation finds its justification in the way FEC-to-label bindings are distributed with LDP[2]. Indeed, two different modes may be used in practice:

(i) In the *ordered LSP control* mode, an LSR associates a label to a prefix only if this prefix is local (typically, the exit point of the LSR), or if it received a label binding for this prefix from its IGP next hop. This mode is thus iterative, as each intermediate LSR waits for a proposal of its downstream neighbor. The LSP is then built from the egress LER to the ingress one. JUNIPER routers use this mode by default, and bind labels to loopback IP addresses only.

(ii) In the *independent LSP control* mode, an LSR generates a label for each prefix appearing in its routing table, local or not. The bindings are then directly distributed to all neighbors.

[2]A description of LDP is available in Section 1.5.4.

This mode does not require any proposal from downstream devices. It is used by default in CISCO routers.

The second mode allows to reduce the time needed to establish an LSP between two LERs. However, it allows LSRs to bind labels to non-local prefixes, and to announce them to all their neighbors without ensuring that the corresponding LSPs are enabled until the exit of the tunnels. However, as mentioned in Section 1.5.5, each MPLS tunnel should end with a standard terminating label (the Implicit or Explicit NULL Label) announced by the egress LER. In the independent LSP control mode, this condition may not be met, and an LSR may thus receive a packet with a label that cannot be associated to an entry in its LFIB[3]. In this case, the router pops the label stack, and continues forwarding the packet based on its IP header. Consequently, the LSP ends abruptly, leading so to an opaque tunnel. A possible scenario with an opaque tunnel is illustrated in Figure 7.1. Note that the device that ends the tunnel may be any LSR in the LSP, and is not necessarily the PH (or the egress LER), as suggested in the model presented in Chapter 2. As CISCO routers use the independent LSP control mode by default, opaque tunnels appear mostly in networks deploying this kind of equipment.

The model for invisible tunnels was already updated in Section 6.3. However, it is valid only if the PH performs PHP. As some operators may decide to enable UHP in their network[4] (in order to ensure some quality of service, for example), this popping mechanism should also be taken into account.

In practice, invisible tunnels behave differently when UHP is enabled, at least when their exit router is a CISCO device running IOS 15.2. Indeed, in this case, when a packet with an IP TTL of 1 is received by the egress LER, it is not dropped, but is directly forwarded to the next hop, without modifying this TTL value. However, the device behaves normally if the IP TTL is greater than or equal to 2 (the TTL is decremented and the packet is transmitted). Consequently, the egress LER is not visible in the `traceroute` output, while its neighbor appears twice. Indeed, the next node answers to two probes, the one that should have expired at the egress LER, and the one sent at the next iteration of `traceroute`. UHP generates thus a particular pattern in a trace, consisting in a duplicate IP address at two successive hops, as illustrated in Figure 7.2.

7.3 TNT: Revealing MPLS Tunnels

This section introduces `TNT` (*Trace the Naughty Tunnels*), a tool able to reveal all MPLS tunnels along a path. It is an extension to `paris-traceroute`, and avoids thus most problems related to load balancing. It has been implemented within `scamper`, and is freely available [55].

[3]The notion of LFIB was introduced in Section 1.5.1.

[4]Around 10% of the operators who took part to the survey described in the previous chapter (see Section 6.1) perform UHP in their network.

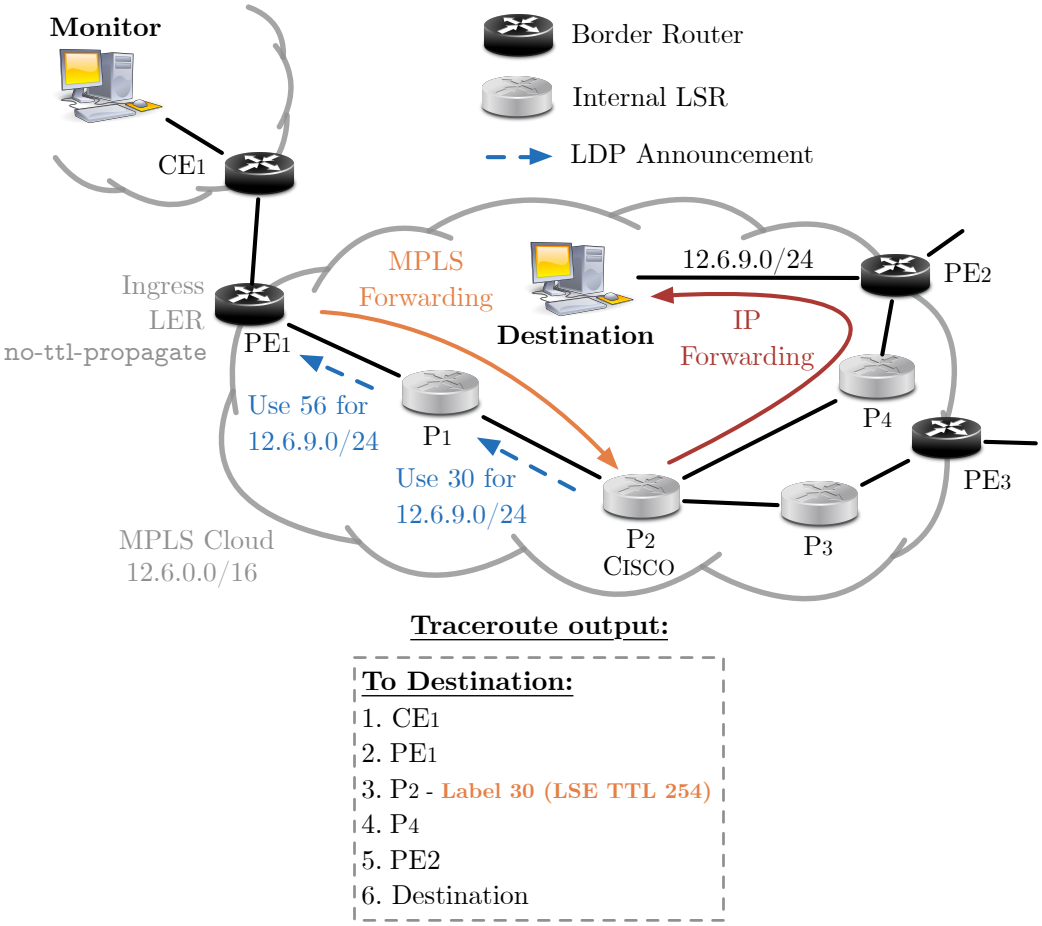
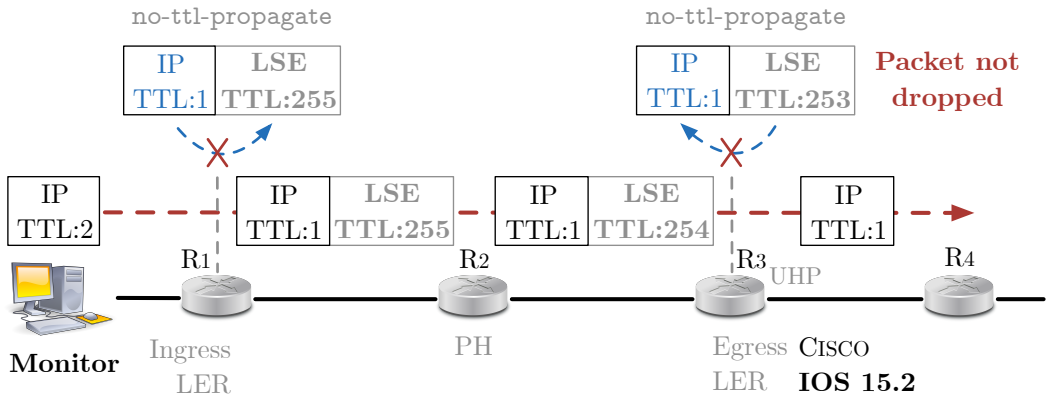
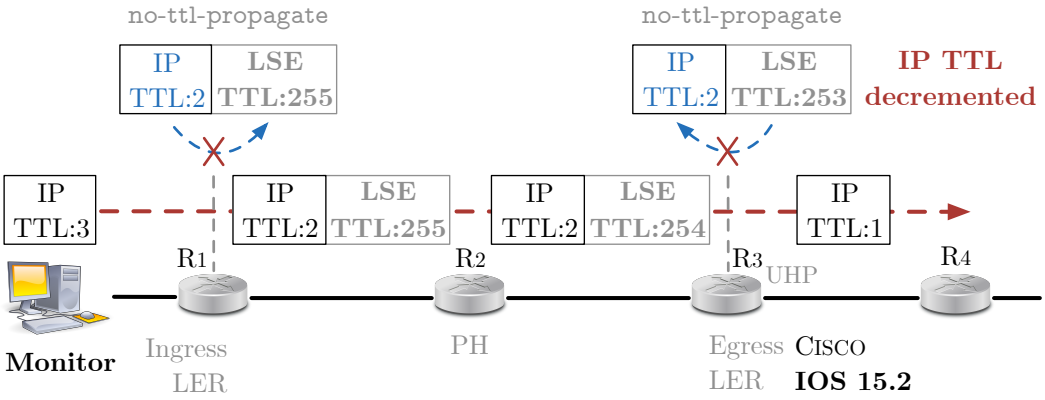


Figure 7.1: Illustration of a network with an opaque MPLS tunnel. Due to the independent LSP control mode, P_2 and P_1 announce a non-local prefix to their upstream neighbor. However, for some reason (e.g., JUNIPER routers in their default configuration, non-MPLS-capable devices, or specific configuration), PE_2 (and P_4) did not bind any label to this prefix. The LSP is then incomplete, and an opaque tunnel arises. Note that opaque tunnels may also appear for transit traffic (i.e., an external prefix) through a specific BGP configuration (see Section 7.3.5).



(a) Second `traceroute` probe.



(b) Third `traceroute` probe.

Taceroute output:

1. R1
2. R4
3. R4
4. ...
5. Destination

R4 appears twice

(c) `traceroute` output.

Figure 7.2: Illustration of the effect of UHP on a `traceroute` output. Two subsequent `traceroute` probes expire at the hop following the egress LER, because packets with an IP TTL equal to 1 are not dropped at the tunnel exit. Remember that when a Cisco device performs UHP, it does not apply Min(IP TTL, LSE TTL), but assumes that its configuration (in terms of TTL propagation) is the same as the one of the ingress LER (see Section 6.2).

```
1  # Define codes for triggers and indicators
2  LSE = 1; LSE-TTL = 2; q-TTL = 3; u-turn = 4;
3  DupIP = 5; RTL = 6; RFPLA = 7;
4
5  trace_naughty_tunnels(target):
6    prev_hop, cur_hop, next_hop = None
7
8    # Run a trace towards the target
9    route = trace(STARTING_TTL, target)
10   # Analyze the trace
11   for (ttl=STARTING_TTL, !halt(ttl, route), ttl++)
12     state, tun_code = None
13     next_hop = route[ttl]
14
15     # Identify tunnels based on indicators
16     tun_code = check_indicators(cur_hop)
17
18     # Run a tunnel revelation, if needed
19     if (tun_code == None)
20       # Check triggers
21       tun_code = check_triggers(prev_hop, cur_hop, next_hop)
22       if (tun_code != None)
23         # Potential hidden tunnel to reveal
24         state = reveal_tunnel(prev_hop, cur_hop, tun_code)
25     elif (tun_code == LSE-TTL)
26       # Potential opaque tunnel to reveal
27       state = reveal_tunnel(prev_hop, cur_hop, tun_code)
28
29     # Output the current hop and any revealed LSP
30     dump(cur_hop, tun_code, state)
31
32     # Sliding pair of IP addresses
33     prev_hop = cur_hop # Candidate ingress LER
34     cur_hop = next_hop # Candidate egress LER
```

Listing 7.1: Pseudo-code for TNT.

7.3.1 Overview

TNT is conceptually illustrated in Listing 7.1. All the operations it performs are enclosed in the function `trace_naughty_tunnels()`. The tool starts by running `paris-traceroute` towards the destination using `trace()`, as shown at line 9. This step allows to collect the intermediate IP addresses on the route between the monitor and the target. Note that `STARTING_TTL` is a

parameter used to avoid tracing repeatedly the nodes close to the source [54].[5] Then, the trace is analyzed hop by hop in order to identify the different MPLS tunnels that might have been crossed by the probes (see line 11). The function `halt()` allows to determine if all hops were considered, and thus, if the loop must be stopped.

At each iteration of the loop, `TNT` first checks whether the current hop (i.e., `cur_hop`) may belong to a visible (i.e., explicit or implicit) or opaque MPLS tunnel based on *indicators* (see line 16). Generally, an indicator is a basic evidence of the presence of these kinds of tunnels, such as a label stack quoted in an ICMP `time exceeded` message (see Section 7.3.2 for details). If `TNT` determines that the hop is part of an explicit or implicit tunnel, no additional probing is required. However, if the tool concludes that it corresponds to the last router of an opaque one, it will try to reveal the hidden part of the LSP, as described by lines 25 to 27. Note that in this case, the ingress LER is the previous hop in the trace.

If `TNT` cannot identify any indicator for the current hop (line 19), it will assume that it may be the egress LER of a potential invisible MPLS tunnel. In this situation, the previous hop (i.e., `prev_hop`) is considered as the corresponding ingress LER. The tool then checks if a *trigger* is positive for the node identified as the tunnel exit (i.e., `cur_hop`), as shown at line 21. In `TNT`, triggers are mainly unsigned values suggesting the possible presence of hidden LSPs through a large path length asymmetry (see Section 7.3.3 for details). If a tunnel is suspected to be hidden between the potential ingress and egress LERs, a revelation is attempted in order to expose its content (see Section 7.3.4 for details). This tasks is performed by the function `reveal_tunnel()`, as shown at line 24. If intermediate hops are found during the process, they are stored in a global stack structure named `revealed_lsrs`.

Finally, at each loop iteration, the collected data is dumped into a warts file, the format used by `scamper` for storing IPv4/IPv6 `traceroute` and `ping` records. This job is performed by the `dump()` function, as shown at line 30. It outputs the trace hop currently analyzed, as well as potential discovered hops (available in the global stack structure `revealed_lsrs`), if any, and additional useful information, such as tags allowing to identify the type of tunnel and the revelation method.

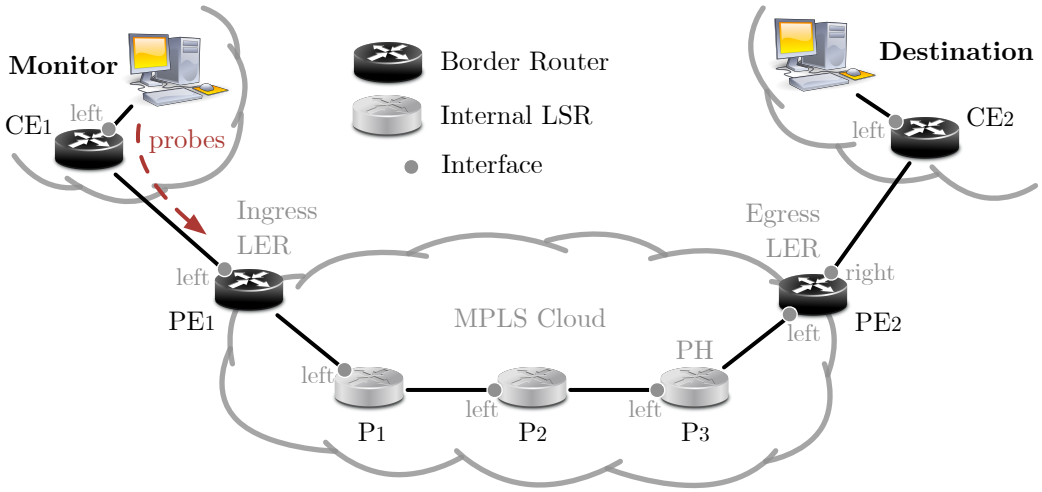
7.3.2 Indicators

As illustrated in Figure 7.3, tunnel indicators allow to retrieve MPLS tunnels (or parts of them) directly from the original `traceroute` output. More specifically, they are evidences of the presence of an explicit, implicit, or opaque LSP. Listing 7.2 describes how `TNT` use them.

Explicit tunnels are identified based on the label stacks directly quoted in ICMP `time exceeded` messages, as shown at lines 7 and 13, and in the `traceroute` output in Figure 7.3.

The indicator for opaque tunnels consists in a single-hop LSP whose unique LSR sent an ICMP `time exceeded` message with a quoted LSE TTL different from 1. As already stated in

[5] This parameter is a command-line argument of `TNT`, and can therefore be set by the user. Its default value is 1.



Traceroute outputs:

Explicit

Hop	IP addr.
1.	CE1.left
2.	PE1.left
3.	P1.left - MPLS
4.	P2.left - MPLS
5.	P3.left - MPLS
6.	PE2.left
7.	CE2.left
8.	Destination

Indicator: LSE

Implicit

Hop	IP addr.	(L_R^T,L_R^P,q-TTL)
1.	CE1.left	(0,0,1)
2.	PE1.left	(1,1,1)
3.	P1.left	(8,2,1)
4.	P2.left	(7,3,2)
5.	P3.left	(4,4,3)
6.	PE2.left	(5,5,1)
7.	CE2.left	(6,6,1)
8.	Destination	(7,7,1)

Indicators: q-TTL & u-turn ($|L_R^T - L_R^P|$)

Opaque

Hop	IP addr. (LSE-TTL)	(L_R^T,L_R^P)
1.	CE1.left	(0,0)
2.	PE1.left	(1,1)
3.	PE2.left - MPLS (252)	(5,5)
4.	CE2.left	(5,5)
5.	Destination	(5,5)

Indicator: $1 << \text{LSE-TTL} < 255$

Figure 7.3: Illustration of TNT's indicators in `traceroute` outputs, where L_R^T and L_R^P represent the lengths of the return paths followed respectively by `traceroute` and `ping` messages. In this simple scenario, forward (i.e., monitor to destination) and return (i.e., destination to monitor) paths are symmetrical. Note that the 3 last values of L_R^T and L_R^P in the opaque case are due to the MIN operation applied at the exit of the return tunnel (see Section 6.2).

```
1  code check_indicators(hop):
2    # The hop must exist
3    if (hop == None)
4      return None
5
6    # Check if a label stack was sent by the hop
7    if (is_mpls(hop))
8      if (Γ_LSE_TTL < hop.lse_ttl < 255)
9        # Opaque tunnel
10       return LSE-TTL
11     else
12       # Explicit tunnel
13       return LSE
14
15   if (hop.qttl > 1)
16     # Implicit tunnel
17     return q-TTL
18
19   # Retrieve key path lengths from raw TTLs
20   L_R^T = path_len(hop.ttl_te)
21   L_R^P = path_len(hop.ttl_er)
22
23   # u-turn is turned into RTL for junos signatures
24   if (|L_R^T - L_R^P| > Γ_u-turn && !signature_is_junos(hop))
25   # Implicit tunnel
26     return u-turn
27
28   return None
```

Listing 7.2: Pseudo-code for checking indicators.

Chapter 2, a value greater than 1 signals the use of the `no-ttl-propagate` option at the ingress LER. This value allows to determine the tunnel length. For example, in Figure 7.3, the LSE TTL received from PE_2 equals 252, which means that the LSP is actually 3 hops long ($255 - 252 = 3$). It is worth noting that the pattern resulting from an opaque tunnel is both an indicator and a trigger. Indeed, `TNT` passively understands that the tunnel is incomplete, and try to reveal its content with new active measurements. The detection of opaque tunnels is performed at lines 7 to 10 in Listing 7.2. Note that algorithm compares the LSE TTL to a minimum threshold value, $\Gamma_{\text{LSE\_TTL}}$, in order to avoid unnecessary revelation attempts (see Section 7.4 for a discussion on the value of this threshold).

Implicit tunnels are detected through the q-TTL and/or u-turn indicators[6]. First, the IP TTL quoted in an ICMP `time exceeded` message (q-TTL) likely reveals the use of the `ttl-propagate`

[6]The q-TTL and u-turn techniques were introduced in Section 2.3.

option at the ingress LER of an MPLS tunnel if its value is greater than 1. Indeed, only the LSE TTL is modified inside an LSP. Consequently, for each subsequent `traceroute` probe expiring inside the LSP, the q-TTL should be one greater, and an increasing sequence of its values should be observed. This indicator is considered at line 15 in Listing 7.2. Second, the u-turn signature relies on the fact that, by default, LSRs send ICMP `time exceeded` messages to the egress LER which, in turns, forwards the packets to the monitor. However, they reply directly to other kinds of probes (e.g., ICMP `echo request`) using their own IP forwarding table, if a route is available. As a result, in general, return paths are shorter for `echo reply` messages than for `time exceeded` ones. Thereby, the u-turn value is the difference between these lengths. It is illustrated in Figure 7.3 (implicit and explicit tunnels follow the same behavior except for the implementation of RFC4950).[7] For example, considering that L_R^T and L_R^P are the lengths of the return paths followed respectively by `traceroute` and `ping` replies[8], the u-turn value on P_1 is obtained computing:

$$\text{u-turn}(P_1) = L_R^T - L_R^P = 8 - 2 = 6$$

Indeed, the ICMP `time exceeded` message crosses 2 LSRs to reach the egress LER, and then 5 devices before arriving at the monitor (L_R^T = 5 + 2 + 1 = 8, the last term taking into account the egress LER), while the `echo reply` packet is directly forwarded via the ingress LER (L_R^P = 2). The u-turn pattern for an LSR P_i in an LSP of length LL can be formalized as follows:

$$\text{u-turn}(P_i) = 2 \times (LL - i + 1) \tag{7.1}$$

Note that in case of PHP, the LH may directly reply to a `traceroute` probe without including the egress LER in the return path. In this situation, the routes followed by the ICMP `time exceeded` and `echo reply` packets are identical, and thus, their lengths too, as shown in Figure 7.3 with P_3. Moreover, in practice, due to the iBGP path heterogeneity (the IGP tie-break rule in particular), the BGP return routes taken by both types of ICMP messages may differ at each intermediate hop. As a result, the u-turn indicator does not necessarily follow exactly Equation 7.1. A small variation may then appear in practice. Consequently, `TNT` compares u-turn values to a specific threshold, $\Gamma_{\text{u-turn}}$, in order to avoid generating too much false positives (see Section 7.4 for more details on this threshold). This operation is performed at line 24 in Listing 7.2

The research of implicit MPLS tunnels must be adapted for JUNIPER routers under Junos. Indeed, it turns out that, by default (i.e., without enabling the `icmp-tunneling` feature, see Section 2.3.2 and `TNT`'s technical report[9] for more details [143, Appendix A.2]), these devices send directly `time exceeded` replies to the source, without forwarding them to the egress LER. Consequently, the u-turn pattern cannot be observed for implicit tunnels composed of this type of equipment. In addition, the u-turn indicator is computed in the same way as the RTL trigger for

[7] Remember that, in this figure, forward and return paths are symmetrical.

[8] The length of the return path can be computed as $L = iTTL - rTTL$ where $rTTL$ is the IP TTL of the ICMP message received by the monitor and $iTTL$ is the corresponding inferred initial TTL (see Section 3.2).

[9] The different simulations on GNS3 were performed by J.-R. Luttringer.

```
1  code check_triggers(prev_hop, cur_hop, next_hop):
2    # Potential ingress and egress LERs must exist
3    if (prev_hop == None or cur_hop == None or prev_hop == cur_hop)
4      return None
5
6    if (cur_hop == next_hop)
7      # UHP invisible tunnel
8      return DupIP
9
10   # Retrieve key path lengths from raw TTLs
11   L_R^T = path_len(cur_hop.ttl_te)
12   L_R^P = path_len(cur_hop.ttl_er)
13   L_F^T = cur_hop.ttl_probe - 1
14
15   if (signature_is_junos(cur_hop))
16     # Router running Junos
17     if (L_R^T - L_R^P ≥ Γ_RTL)
18       # PHP invisible tunnel
19       return RTL
20   else
21     # Router running another OS than Junos
22     if (L_R^T - L_F^T ≥ Γ_RFPLA)
23       # PHP invisible tunnel
24       return RFPLA
25
26   return None
```

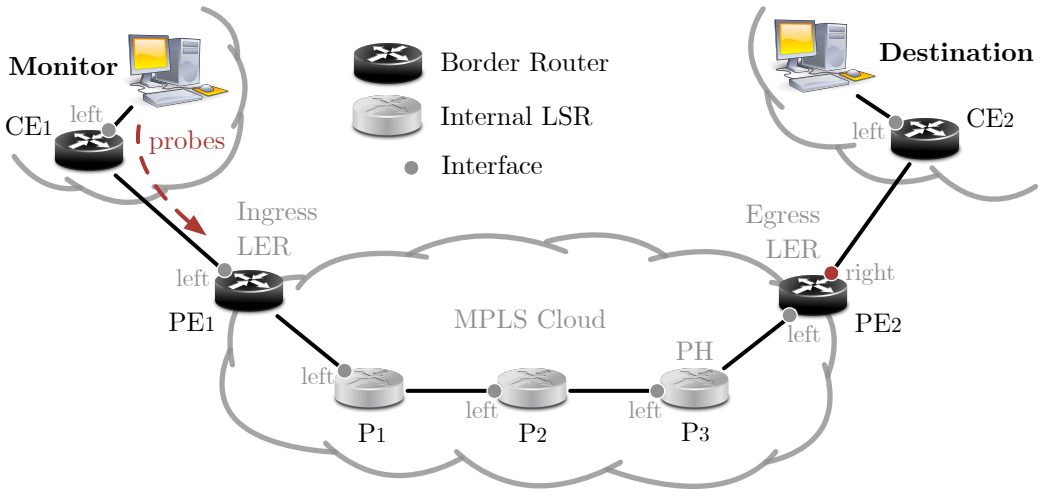
Listing 7.3: Pseudo-code for checking triggers.

Junos routers[10] (see Section 7.3.3). Therefore, to avoid any confusion, TNT introduces an exception for them, as shown at line 24 in Listing 7.2. The tool considers first the path length difference as a trigger, and then falls back to an indicator if the revelation attempt fails (not shown in Listing 7.1 for clarity). Finally, note that when `icmp-tunneling` is enabled, `time exceeded` replies start with a TTL of 254 instead of 255. As a result, compared to CISCO routers, a bigger difference is observed when computing the path length difference with `echo request` replies.

7.3.3 Triggers

Indicators are evidences of the presence of MPLS tunnels that prevent TNT from firing new probes (with the exception of opaque tunnels). On their side, triggers are unsigned values or patterns that suggest the existence of invisible tunnels (PHP or UHP) whose content could be revealed using additional measurements (see Section 7.3.4). They are illustrated based on `traceroute`

[10]The RTL trigger is only computed for routers with signature $< 255, 64 >$, i.e., JUNIPER devices running Junos.



Traceroute outputs:

PHP Invisible:

Hop	IP addr.	(L_R^T,L_R^P[|$L_R^{P,J}$])
1.	CE1.left	(0,0)
2.	PE1.left	(1,1)
3.	PE2.left	(5,5|2)
4.	CE2.left	(5,5|3)
5.	Destination	(5,5|4)

Triggers:
RFPLA (L_R^T - L_F^T)
RTL (L_R^T - $L_R^{P,J}$)

UHP Invisible:

Hop	IP addr.	(L_R^T,L_R^P)
1.	CE1.left	(0,0)
2.	PE1.left	(1,1)
3.	CE2.left	(3,3)
4.	CE2.left	(3,3)
5.	Destination	(4,4)

Trigger: DupIP

Figure 7.4: Illustration of TNT's triggers in `traceroute` outputs, where L_R^T and L_R^P represent the lengths of the return paths followed respectively by `traceroute` and `ping` messages, and L_F^T the length of the forward route taken by the `traceroute` probes (L_F^T = Hop - 1). Note that $L_R^{P,J}$ identifies the specific case of JUNIPER routers running Junos. In this simple scenario, forward and return paths are symmetrical.

outputs in Figure 7.4, while Listing 7.3 show how they are checked by TNT on a potential egress LER.

First, the tool searches for a duplicate IP address that may signal an invisible UHP tunnel (line 6 in Listing 7.3). As explained in Section 7.2, this pattern suggests that the egress LER is a CISCO router running IOS 15.2. Indeed, when UHP is enabled, this device forwards directly to the next hop any packet received with an IP TTL of 1, without decrementing this TTL. Consequently, it is not visible in the trace, while, on the contrary, its neighbor (CE_2 in Figure. 7.4) appears twice (duplicate IP address in the output).

The two remaining triggers, RFPLA (*Return/Forward Path Length Asymmetry*) and RTL (*Return Tunnel Length*), were already introduced in Section 6.4.1. As a quick reminder, they infer the presence of PHP invisible tunnels by comparing the lengths of different paths. More precisely, RFPLA is the difference between the return and forward path lengths for `traceroute` (i.e., L_R^T- L_F^T), while RTL is the difference between the lengths of the return paths followed by the `time exceeded` and `echo reply` messages (i.e., L_R^T- L_R^P). Both triggers estimate the number of LSRs in the tunnel on the return path, exploiting the MIN(IP TTL, LSE TTL) operation[11] performed at its exit. As stated in the previous chapter, RFPLA is subject to BGP path asymmetry, and so, to false positives and negatives. On its side, RTL is only sensitive to ECMP. In the absence of an invisible tunnel, those triggers should have a value equal to or close to 0. Any significant deviation from this value is therefore interpreted as the potential presence of an invisible MPLS cloud, and brings TNT to run additional revelation techniques (see Section 7.3.4). As explained in Section 6.4.1, RTL works only with JUNIPER routers under Junos, while RFPLA is more generic. However, as RTL is more accurate, it should be applied in priority, when possible. Therefore, TNT uses the network fingerprinting technique introduced in Chapter 3 to determine the router brand of the potential egress LER (line 15 in Listing 7.3).

Both triggers are evaluated based on the three key distances obtained thanks to the IP TTLs in the reply messages received by the monitor (lines 11 to 13 in Listing 7.3). In order to take into account ECMP and BGP path asymmetry, TNT compares the values of RFPLA and RTL to the pre-defined minimum thresholds Γ_{RFPLA} and Γ_{RTL} (lines 17 and 22 in Listing 7.3). Indeed, a trigger with a value different from 0 does not necessary imply the presence of an invisible tunnel. In this way, a tradeoff can be found between the number of tunnels revealed by the tool, and the additional measurement cost induced by the discovery techniques. Note that the calibration of TNT, and more specifically, the values of the different thresholds, will be discussed in Section 7.4.

It is worth noting that an invisible shadow effect applies for RTL and RFPLA after a PHP invisible tunnel. Indeed, the triggers remain positive for a few nodes after the egress LER due to the MIN(LSE TTL, IP TTL) behavior. This effect is visible in Figure 7.4: RFPLA (CE_2) = 2, RFPLA (destination) = 1, etc. It applies for a number of hops equal to the length of the hidden LSP minus 1. These shadows are the reasons why TNT does not look for consecutive invisible

[11] This operation was explained in Section 6.2.

tunnels in a trace. Note that, in case of UHP, the MIN operation is not applied on the return path, and only the duplicate IP address is visible.

Threshold calibration will be discussed in detail in Section 7.4. The optimal parameter values can provide a 80/20% success/error rates, errors being due to the BGP and ECMP noises. Moreover, the order in which TNT considers indicators and triggers reflects their reliability, and so, their respective success rates. In the tool, a code is assigned to each of them, as shown at the beginning of Listing 7.1. A lower code number represents a higher priority, and a higher revelation success rate. Thus, if a hop matches simultaneously multiple triggers (RTL and RFPLA for example), it is tagged with the one having the highest priority (i.e., RTL in our example).

7.3.4 Hidden Tunnel Revelation

Listing 7.4 gives a simplified view of TNT's tunnel revelation. The first step consists in launching a regular `traceroute` towards the candidate egress LER[12] (line 7). In the algorithm, `REV_STARTING_TTL` is the starting TTL used for the revelation, which corresponds to 2 hops before the candidate ingress hop. During this first attempt, TNT may fail to reach the candidate egress LER (line 10), and/or to cross again the candidate ingress LER (line 13)[13]. Otherwise, TNT tries to expose the content of a tunnel, using several other traces if needed, and four additional output states are possible:

(i) an LSP composed of at least 2 LSRs is discovered in the trace towards the candidate egress LER (line 17). It is a *Direct Path Revelation* (*DPR*)[14];

(ii) an LSP having more than one LSR is revealed using several iterations (line 45). It is a *Backward-Recursive Path Revelation* (*BRPR*);

(iii) nothing is revealed, the candidate ingress and egress nodes are still consecutive hops in the first discovery trace (line 39);

(iv) only one LSR is discovered (line 42), although several iterations have been tried. The LSP contains a single internal node (*1Hop-LSP*). DPR and BRPR cannot be distinguished in this case.

In Listing 7.4, `extract_hop()` (line 19) identifies the IP address revealed between the ingress and egress LERs, while `trace_hop()` (line 36) runs `traceroute` towards the target, and extracts any discovered IP address. Both methods return the target if they do not find any new IP hop.

When a tunnel performing UHP ends with a CISCO device running IOS 15.2, an additional test, called *buddy test* (line 27), is required to retrieve the outgoing IP interface of the egress LER (the right interface of PE_2, in red in Figure 7.4). Indeed, a probe towards this interface can

[12]The term *candidate* is used here because, at this point, there is no certainty that a tunnel is hidden there.
[13]This condition is necessary in order to follow the potential invisible LSP.
[14]DPR and BRPR were described in detail in Section 6.4.2.

```
state reveal_tunnel(ingress, egress, tun_code):
  if (ingress == None or egress == None)
    return None
  buddy_bit = False
  # Regular traceroute towards the candidate egress
  target = egress
  route = trace(REV_STARTING_TTL, target)
  if (last_hop(route) != egress)
    # The target did not respond (revelation not possible)
    return "TARGET-NOT-REACHED"
  else if (ingress ∉ route)
    # The forwarding path differs (revelation not possible)
    return "INGRESS-NOT-FOUND"
  else if (distance(ingress, egress, route) > 1)
    # Multiple LSRs revealed with DPR
    push_lsp_to_revelation_stack(ingress, egress, route)
    return "DPR"
  else
    revealed_ip = extract_hop(ingress, egress, route)
    for iTR=0;;
      if (revealed_ip == target)
        if (tun_code != DupIP || buddy_bit)
          # No more progression in the revelation
          break
        else
          # Try with the buddy for the DupIP trigger
          target = buddy(revealed_ip)
          buddy_bit = True
      else
        # A new hop has been revealed
        iTR++
        push_hop_to_revelation_stack(revealed_ip)
        target = revealed_ip
        buddy_bit = False
      # Run traceroute towards new target to reveal another hop
      revealed_ip = trace_hop(REV_STARTING_TTL, ingress, target)
    if (iTR == 0)
      # No revelation (fail)
      return "NOTHING-REVEALED"
    if (iTR == 1)
      # Single-hop LSP revealed (cannot distinguish DPR from BRPR)
      return "1HOP-LSP"
    else
     # Hop by hop revelation with BRPR
     return "BRPR"
```

Listing 7.4: Pseudo-code for revealing invisible tunnels.

force a reply from the incoming one (the left interface of PE_2 in Figure 7.4), that can then be used to reveal the tunnel content. The function `buddy()` is in charge of determining this outgoing interface. To do so, it assumes a point-to-point connection between the egress LER and the next hop. The IP addresses on this point-to-point link are called *buddies*. In most cases, they belong to a /31 or a /30 prefix [107]. The test may require an additional `ping` to infer the right prefix.

The algorithm allowing to identify the buddy (IP_B) of a given IP address (IP_T) is quite simple. Indeed, a /30 prefix gathers 4 possible IP addresses. Let us number them from 0 to 3. In such a prefix, addresses 0 and 3 are respectively the network and broadcast addresses, and are therefore not assigned to an interface. Hence, if IP_T is address 0 or 3, the function `buddy()` concludes that the point-to-point link uses a /31 prefix, and IP_B is address 1 or 2 respectively. On the contrary, if IP_T is address 1, a `ping` is run towards address 0. If the monitor gets a reply, the address is allocated to a device, and `TNT` infers that the buddies belong to a /31 prefix. Consequently, IP_B is address 0. However, if no ICMP `echo reply` message is received, the prefix is probably a /30 one, and IP_B is assumed to be address 2. The same reasoning is applied if IP_T is address 2.

Note that `TNT` uses UDP probes when targeting the outgoing interface of the egress LER in order to force the device to generate an ICMP `destination unreachable` message. Such an error message, as `time exceeded` ones, allows to get the incoming interface of the targeted router, in opposition to `echo reply` packets that are built with the destination of the probe as source IP address.

Once the incoming interface of the egress LER (i.e., the left interface of PE_2 in Figure 7.4) has been identified, the content of the tunnel can be revealed. Two scenarios are possible:

(i) UHP is applied for transit traffic only. If MPLS is also used for all the internal IGP prefixes with PHP, BRPR is able to expose the tunnel content. If MPLS does not forward the traffic to internal destinations, DPR can reveal the hidden routers.

(ii) UHP is used over the whole MPLS domain, even for internal destinations. In this case, the function `buddy()` must be applied iteratively in order to discover all internal LSRs.

Finally, it is worth noting that BRPR also works natively for CISCO devices running IOS 12.4, and performing UHP (i.e, without calling `buddy()`). Indeed, this kind of equipment does not forward packets with an IP TTL of 1 to the next hop. As a result, the egress LER appears directly in the `traceroute` outptut, and the revelation technique can be applied in the same way as for tunnels performing PHP.

7.3.5 Reproducibility and Practical BGP Configurations

The measurement techniques used in `TNT` were validated thanks to GNS3[15]. First, this emulator allowed to verify that the different inference assumptions made based on observations in real

[15]This validation was performed by J.-R. Luttringer. More details on the simulations can be found in `TNT`'s technical report [143].

networks were correct, and reproducible in a controllable environment. Second, some of the phenomena exploited by the tunnel revelation methods were discovered in the testbed. For example, the TTL processing of some common OSs run by many real routers (the POP operation in particular) could be reverse-engineered using the emulation tool. Finally, GNS3 allowed to test TNT's features in a controllable environment during its development. Generally speaking, the aim of this validation step was to reproduce all common behaviors observed in real networks, and, on the opposite, verify that the different basic behaviors set up in the virtual environment, based on standard MPLS and BGP behaviors, could be retrieved in the Internet.

In practice, four distinct router OSs were considered: two standard CISCO OSs (IOS 12.4 and 15.2), and two virtualized versions of Junos (Olive and VMX, the only JUNIPER OSs that could be emulated successfully within GNS3). The different tests performed during this validation should be representative of most behaviors existing in real networks.

The topology presented in Figure 7.5 was reproduced in the virtual environment. The two LERs were considered as *Provider Edge (PE)* routers in an ISP, i.e., border routers that run (e)BGP sessions. Two main BGP configurations allowed the MPLS tunneling of transit traffic. In the first one, the BGP next hop was the loopback IP address of the PE itself (thanks to the `next hop self` command). In the second, used with opaque tunnels only, the BGP next hop was the eBGP neighbor. In this last situation, the prefix of the connected subnet (or the IP address) had to be redistributed in the ISP network. In both cases, LDP was used as label distribution protocol, and a label was associated to the BGP next hop at each ingress LER.

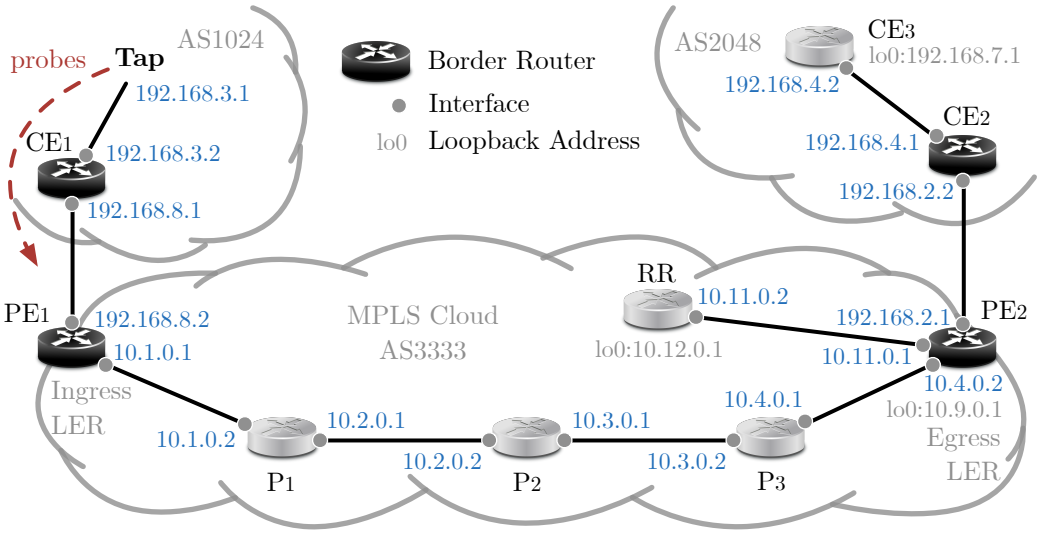


Figure 7.5: Topology for GNS3 emulations, where RR is a BGP route reflector. An MPLS tunnel is established between PE_1 and PE_2.

Only simple structures were considered, homogeneous in terms of OSs and BGP/MPLS/LDP configurations, at the domain scale. The two modes of LDP were investigated separately during

the tests. All connected IGP prefixes were announced in the independent LSP control mode (default mode with CISCO), while only loopback addresses were mapped to labels in the ordered LSP control mode (default mode with JUNIPER). In the simulations, forward and return MPLS tunnels had a symmetrical configuration. All LSRs also had a global routing table thanks to a route reflector or a redistribution in the IGP routing control plane. The devices could therefore answer natively to ping requests. The AS-scale BGP prefix was advertised using a global aggregation. The BGP inter-domain link was managed by the neighbor, but could be redistributed in IGP as a connected one.

The configurations of the virtual routers in the case of an invisible MPLS tunnel performing PHP at its exit is available in Listing 7.5. The operating system running on each device was CISCO IOS 15.2. The corresponding output of `TNT` targeting the loopback address of CE_3 can be found in Listing 7.6. The tool revealed three invisible hops (H_1 to H_3) corresponding to the three devices between PE_1 and PE_2. The trigger that highlighted the presence of the tunnel was RFPLA, and the hidden nodes were revealed by BRPR in three recursive steps. Note that RFPLA values different from 0, and the IP TTL values of the probes quoted inside the ICMP responses (q-TTL), are shown in the output.

One of the most surprising behaviors observed in the Internet is the one resulting from the opaque tunnel shown in Figure 7.3. It is intriguing, especially at the BGP scale, as it implies a badly-controlled tunnel ending. This kind of MPLS tunnel is the only one that required a modification of the BGP configuration to show up. Indeed, the next hop self feature had to be disabled, and the loopback address of the neighbor had to be selected as the BGP next hop using the command `neighbor <IP> ebgp-multihop <#hops>`[16]. Then, this IP address was redistributed via a static route within the IGP.

While DPR is expected to work mostly with JUNIPER devices in their default configuration, the technique could be applied in a domain composed of CISCO devices. Indeed, these routers could be set up in order to use MPLS for transit traffic only (i.e., injecting only loopback addresses into LDP) thanks to the command `mpls ldp label allocate global host-routes`. In this case, DPR was able to reveal the hidden LSRs.

GNS3 allowed to discover many different behaviors associated to UHP (only tested on CISCO devices and LDP). First, as already stated in the previous section, devices under IOS 12.4 do not generate the pattern of the duplicate IP address. The egress LER appears directly in the `traceroute` output, with the Explicit NULL Label. Second, the command `mpls ldp label allocate global host-routes` allows DPR to reveal the internal LSRs of UHP tunnels (`buddy()` is still necessary to reveal the egress LER with IOS 15.2). This case seems the most frequent in real networks, even more than the default CISCO configuration that requires BRPR with the buddy for each hop. Finally, `TNT` identified a more sophisticated pattern during measurements in the Internet: BRPR working without the use of the buddy. This behavior could

[16] In this command, IP is the address of the neighbor's loopback address, and #hops is the maximum distance of the eBGP session expressed as a TTL value.

```
PE1
version Cisco IOS 15.2
mpls label protocol ldp
no propagate-ttl
router bgp 3333
 redistribute connected
 redistribute ospf 10
 neighbor 10.12.0.1 remote-as 3333
 neighbor 10.12.0.1 next-hop-self
 neighbor 192.168.8.1 remote-as 1024
 neighbor 192.168.8.1 next-hop-self

PE2
version Cisco IOS 15.2
mpls label protocol ldp
no propagate-ttl
router bgp 3333
 redistribute connected
 redistribute ospf 10
 neighbor 10.12.0.1 remote-as 3333
 neighbor 10.12.0.1 next-hop-self
 neighbor 192.168.2.2 remote-as 2048
 neighbor 192.168.2.2 next-hop-self

P1
version Cisco IOS 15.2
mpls label protocol ldp
no propagate-ttl
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333

P2
version Cisco IOS 15.2
mpls label protocol ldp
no propagate-ttl
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333

P3
version Cisco IOS 15.2
mpls label protocol ldp
no propagate-ttl
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333
```

Listing 7.5: Cisco router configurations for an invisible MPLS tunnel with PHP in GNS3.

```
trace [icmp-paris] to 192.168.7.1
  1 192.168.3.2 7.520 ms <255,255> qttl=1
  2 192.168.8.2 29.927 ms <254,254> qttl=1 [MPLS,INV,ING,RFPLA]
 H1 10.1.0.2 50.051 ms <253,253> qttl=1 [MPLS,LSR,RFPLA,BRPR,step=3]
 H2 10.2.0.2 60.102 ms <252,252> qttl=1 [MPLS,LSR,RFPLA,BRPR,step=2]
 H3 10.3.0.2 59.876 ms <251,251> qttl=1 [MPLS,LSR,RFPLA,BRPR,step=1]
  3 10.4.0.2 80.380 ms <250,250> qttl=1 rfpla=3 [MPLS,INV,EGR,RFPLA]
  4 192.168.2.2 69.890 ms <250,250> qttl=1 rfpla=2
  5 192.168.4.2 99.833 ms <250,250> qttl=1 rfpla=1
```

Listing 7.6: Output of TNT running over an invisible MPLS tunnel performing PHP. The topology of the virtual network is available in Figure 7.5, while the router configurations can be found in Listing 7.5. Note that this output corresponds to a development version of TNT, and may slightly differ from the output of the last version of the tool.

be reproduced in the virtual environment using the command `mpls ldp explicit-null [for prefix-acl]` in order to force UHP proposals (i.e., the announcements of the Explicit NULL Label) for the loopback addresses of the egress LERs only.

Finally, note that some heterogeneous configurations were also evaluated. They allowed to discover many intriguing corner cases that are discussed in TNT's technical report [143, Appendix]. In short, some of these configurations caused an incorrect TTL processing, and others hid even more the tunnel to TNT. In some rare cases, a brute-force mode[17] was necessary to expose the tunnel content.

7.4 TNT: Calibration and Probing Cost

Section 7.3 showed that TNT mainly relies on four parameters when searching for tunnel indicators and triggers. These parameters are $\Gamma_{\text{LSE\_TTL}}$ and $\Gamma_{\text{u-turn}}$ for opaque and implicit tunnels respectively, and Γ_{RTL} and Γ_{RFPLA} for invisible ones. Their values have an impact on the performance of the tool, in terms of number of tunnels discovered, and probing cost. Indeed, if they are too high, TNT may fail to detect some tunnels. On the contrary, if they are too low, the tool may overload the network with its probes. Therefore, they must be chosen properly. This section aims at calibrating TNT's parameters (Section 7.4.2), as well as evaluating the cost associated to the tunnel revelation (Section 7.4.3).

[17] In brute-force mode, TNT tries to reveal a tunnel for each pair of IP addresses, without considering the values of the triggers.

7.4.1 Measurement Setup

TNT was deployed on three Archipelago monitors [38] located in Europe (Belgium), North America (San Diego), and Asia (Tokyo). The tool was run on April 6th, 2018 towards a set of 30,000 destinations randomly chosen in the list of targets used by the measurement platform, and evenly distributed amongst the three measurement points. The parameter `STARTING_TTL` (see Listing 7.1) was set to 1 in order to consider the worst case in terms of probing cost, and to detect potential tunnels close to the monitors.

By design, u-turn is equivalent to RTL for JUNIPER routers running Junos, as shown at lines 24 and 17 in Listings 7.2 and 7.3 respectively. However, in practice, $\Gamma_{\text{u-turn}}$ does not have the same value as Γ_{RTL}. Indeed, $\Gamma_{\text{u-turn}}$ equals 0 because, for CISCO routers, a path longer for ICMP `time exceeded` messages than `echo reply` ones indicates a different routing behavior for each kind of replies, and thus, an implicit MPLS tunnel. In the implementation of TNT, this condition is reinforced by looking for at least two consecutive hops having a cumulated u-turn value greater than or equal to 3.

Besides, the measurements performed on Archipelago showed that abnormal[18] LSE TTL values oscillate between 236 and 254, the main proportion being located between 249 and 254, as shown in Figure 7.6. It suggests thus that, in the majority of the cases, opaque LSPs are rather short. Consequently, a value of 236 for $\Gamma_{\text{LSE\_TTL}}$ is enough for detecting the presence of opaque tunnels.

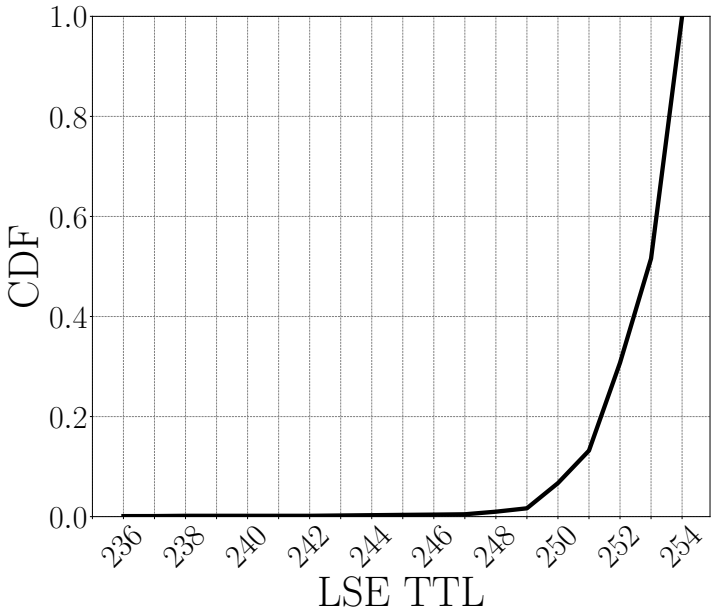


Figure 7.6: Distribution of LSE TTL values different from 1 in the data collected on Archipelago.

[18]Abnormal means here different from 1, the LSE TTL value that should be obtained in ICMP `time exceeded` messages.

In order to perform the calibration of Γ_{RTL} and Γ_{RFPLA}, a variation of their values was considered. Each of them ranged from 0 to 4. A full measurement campaign was conducted for each pair of threshold values (thus, a total of 25 configurations). `TNT` was run in brute-force mode in the different campaigns. It means that, even when no trigger or indicator was observed for a pair of candidate LERs, a tunnel revelation was attempted. The collected data was used as a basis to evaluate the quality and cost of each pair of threshold values.

7.4.2 Calibration

With the help of well calibrated thresholds, RTL and RFPLA are able to predict if a hidden LSP may exist between two successive hops in a trace. On their side, DPR and BRPR are two revelation techniques that can directly determine if invisible LSRs interconnect these hops. With that in mind, one can assess the performance of RFPLA and RTL triggers through the analysis of *True Positive Rate (TPR)* and *False Positive Rate (FPR)*. They are plotted on a *Receiver Operating Characteristic (ROC)* curve available in Figure 7.7. In this analysis, each pair of candidate LERs was associated to one of the following classes:

(i) *True Positive (TP)*: RTL or RFPLA was positive on the candidate egress LER, and a hidden LSP was revealed between the two nodes.

(ii) *False Positive (FP)*: RTL or RFPLA was positive on the candidate egress LER, but no LSR was discovered between the two nodes.

(iii) *True Negative (TN)*: both triggers were negative on the candidate egress LER, and a brute-force revelation attempt did not reveal any hidden MPLS router between the two nodes.

(iv) *False Negative (FN)*: both triggers were negative on the candidate egress LER, but a brute-force revelation attempt exposed a hidden LSP between the two nodes.

Then, considering that $|X|$ represents the number of elements in class X, TPR and FPR could be easily computed as follow:

$$TPR = \frac{|TP|}{|TP| + |FN|} \tag{7.2}$$

$$FPR = \frac{|FP|}{|FP| + |TN|} \tag{7.3}$$

The revelation techniques were considered as reliable during the analysis. It means that if DPR and BRPR did not reveal any LSR between two nodes, no tunnel was assumed to exist there. Note that inconclusive cases, corresponding to the states INGRESS-NOT-FOUND and TARGET-NOT-REACHED (see Listing 7.4), were not taken into account.

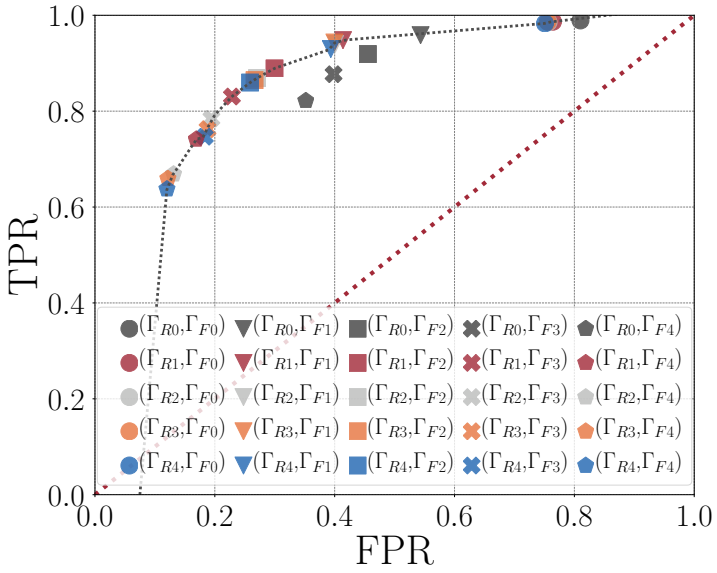


Figure 7.7: ROC curve providing the efficiency of TNT according to different values of Γ_{RFPLA} and Γ_{RTL}. In the legend, Γ_{Rx} refers to Γ_{RTL} with the value x, while Γ_{Fy} identifies Γ_{RFPLA} with the value y. The red dotted diagonal provides the separation between positive (above part of the graph) and negative (below part of the graph) results for TNT. The black dotted line is the extrapolation of the measurement values (excluding Γ_{R0} values which appear as being outliers, as expected).

The ROC curve was obtained by varying Γ_{RTL} and Γ_{RFPLA} between 0 and 4. We can observe that the results are essentially positive for TNT. Some threshold values, (Γ_{R1}, Γ_{F3})[19] and (Γ_{R_2}, Γ_{F_3}), are even reasonably close to the perfect case (upper left corner). They are thus considered as the best choices for defining Γ_{RTL} and Γ_{RFPLA}, and allow a compromise close to 80%-20% for TNT. It means that, with this calibration, the tool is able to reveal at least 80% of existing hidden MPLS tunnels, while sending useless revelation probes for 2 real IP links out of ten.

7.4.3 Probing Cost

Figure 7.8 illustrates the probing cost associated to TNT. In particular, it focuses on the additional measurements triggered by RTL and RFPLA for discovering invisible tunnels. The light grey zone (labeled as "Original") corresponds to the probes sent by the regular `traceroute`[20]. The blue, orange, and dark grey regions represent the ones sent by the tunnel revelation techniques. In particular, the blue area gathers the additional measurements that allowed to discover the content of an invisible tunnel. On the contrary, the orange zone refers to the ones that did not expose any hidden LSP. Finally, the dark grey region refers to inconclusive revelations for which the triggers signaled the potential presence of a tunnel, but TNT was unable to reach the

[19] Γ_{Fx} and Γ_{Rx} represent respectively Γ_{RFPLA} and Γ_{RTL} with the value x.
[20] This `traceroute` is the first run by TNT in order to obtain the IP path between the monitor and the destination.

candidate egress LER (generally due to an unresponsive IP interface), or to cross again the candidate ingress LER (ECMP or BGP routing noises because the destination changed), as shown at lines 8 to 13 in Listing 7.4.

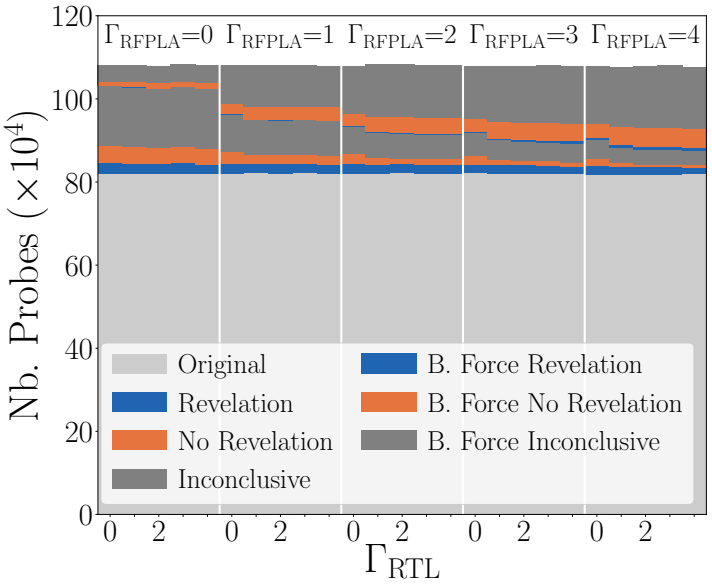


Figure 7.8: Probing cost associated to TNT according to Γ_{RFPLA} and Γ_{RTL} thresholds.

Focusing first on the standard mode of TNT (i.e, non-dashed areas), the amount of probes sent in order to expose the content of actual invisible tunnels (blue) remains almost stable, whatever the values of Γ_{RFPLA} and Γ_{RTL}. However we can observe a very slow decrease. Less tunnels are revealed when the thresholds have high values. Further, the additional traffic generated by erroneous estimates (orange), or by inconclusive revelations (dark grey), clearly decreases while Γ_{RFPLA} increases. This result is aligned with Section 7.4.2 that highlighted that the best values for this threshold are between 2 and 3. Note that RFPLA is more generic, but less reliable than other triggers. On the contrary, Γ_{RTL} has a minor effect on the amount of probes sent, because it is specific to JUNIPER devices, and more accurate.

Then, the hatched zones (orange, dark grey, and blue) allow to compare TNT's brute-force mode to its standard one. First, the amount of probes sent increases with Γ_{RFPLA} (the impact of Γ_{RTL} is quite negligible) in case of brute force. The same conclusion applies for the number of inconclusive revelations. Second, the quantity of additional measurements that exposed an invisible tunnel is really low in comparison to the standard mode.

Generally speaking, we can observe that the overhead of TNT is quite limited compared to a basic active measurement campaign, taking into account the value of the information it brings. In particular, if the tool is calibrated with the right threshold values (e.g., Γ_{R_1}, Γ_{F_3}), it generates less than 10% of additional probes compared to a regular `traceroute` campaign, and reaches a satisfying performance where 80% of the tunnels are revealed.

7.5 TNT: Tunnels Quantification

This section aims at discussing how TNT and its features behave in the wild, as well as the current state of MPLS deployment in the Internet. In particular, it analyzes the success rate of each indicator and trigger with respect to the revelation techniques, quantifies the tunnels discovered with the tool, and studies some of their basic characteristics.

7.5.1 Measurement Setup

TNT was deployed on the Archipelago infrastructure on April 23rd, 2018 with the thresholds Γ_{RFPLA} and Γ_{RTL} set respectively to 3 and 1, according to the results discussed in Section 7.4.2. The parameter STARTING_TTL (see Listing 7.1) was set to 1. The tool was run on 28 monitors scattered all around the world: Europe (9), North America (11), South America (1), Asia (4), and Australia (3). The overall set of destinations, nearly 2,800,000, was inherited from Archipelago's target list, and spread over the set of measurement points in order to speed up the probing process.

A total of 522,049 distinct IP addresses (excluding traceroute targets) were collected, of which 28,350 were not publicly routable, and therefore excluded from the dataset. Each routable address was pinged once per monitor, for fingerprinting.

7.5.2 Results

Table 7.1 provides the amount of probes sent by traceroute, ping, and buddy(). The row "original" refers to measurements not run by tunnel revelation techniques.

Status		**Number of probes**		
		traceroute	ping	buddy()
	original	63,559,385	7,109,075	–
Attempt	revelation	2,190,275	206,842	19,181
	no revelation	1,640,224	–	556
	TARGET-NOT-REACHED	4,174,404	–	9,888
	INGRESS-NOT-FOUND	1,790,900	–	7,326

Table 7.1: Raw number of probes sent by TNT over the set of 28 monitors.

The main result from this table is the amount of probes involved in inconclusive revelations, split between TARGET-NOT-REACHED (TNT was unable to reach the potential egress LER) and INGRESS-NOT-FOUND (the revelation probes did not cross the candidate ingress LER). In particular, twice more probes did not reach their destination in comparison to the ones that allowed to expose hidden tunnels. A possible cause may be an ICMP rate limiting applied by some devices. Another explanation is that some routers may have replied to initial traceroute probes

with an IP address that is not globally announced. Consequently, additional probing towards these addresses failed, as no route was available to reach them.

Figure 7.9 shows, for each monitor, the proportion of paths that cross at least one MPLS tunnel. For around 75% of the monitors, at least one route out of two contains MPLS routers. The technology is therefore well present in the Internet today, and its possible impact on network measurements should not be underestimated. Note that this figure can be compared with Figure 3.5 that shows the same results, but for explicit tunnels only. The proportion of paths is larger in Figure 7.9, as TNT takes into account all types of MPLS tunnels. Moreover, the data used to plot the graph in Chapter 3 was collected in 2013, meaning 5 years ago. The deployment of MPLS has therefore probably increased in the meantime.

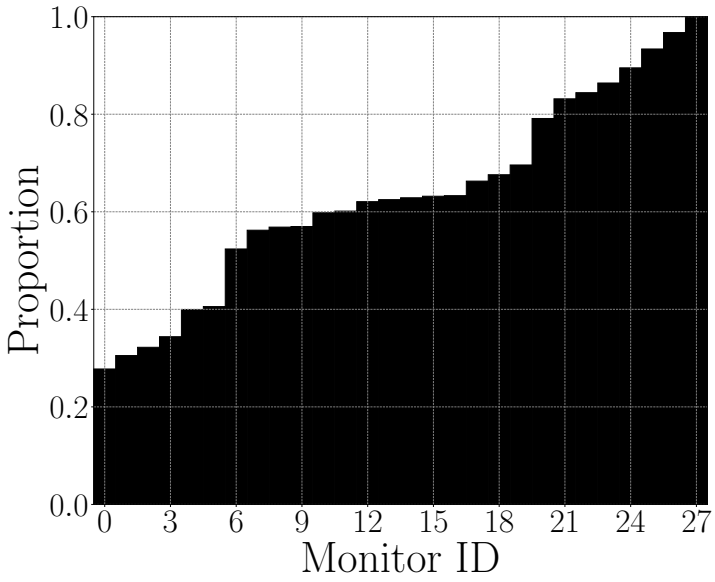


Figure 7.9: Proportion of paths, per monitor, crossing at least one MPLS tunnel.

Table 7.2 provides the number of MPLS tunnels discovered by TNT. They are classified according to their type, as indicated by the first column. The indicators and triggers are provided, as well as the revelation technique, if any. Without any surprise, explicit tunnels are the most present (76% of the identified tunnels).

Implicit tunnels represent 5% of the whole dataset, and are more often identified based on the u-turn indicator than the q-TTL one. However, those results must be taken with caution, as u-turn is subject to false positives (see Section 7.3.2), in opposition to q-TTL, which is more reliable [44]. The quantity of implicit tunnels is therefore probably overestimated.

Opaque tunnels are less prevalent, and correspond to 1.7% of the tunnels that were discovered. This observation confirms previous empirical results [53], and was somewhat expected, as this kind of tunnel is due to a particular label distribution within MPLS clouds composed of CISCO routers (see Section 7.2). It is worth noting that the revelation techniques do not perform well on

Tunnel Type	Indicator	Trigger	Revelation Technique				#Tunnels
			DPR	BRPR	1Hop-LSP	Mix	
Explicit	LSE	–	–	–	–	–	150,036
Implicit	q-TTL	–	–	–	–	–	2,689
	u-turn	–	–	–	–	–	7,216
Opaque	LSE-TTL		22	17	43	–	3,346
Invisible PHP	–	RTL	11,268	1,191	2,595	279	15,333
	–	RFPLA	5,903	2,555	3,260	1,012	12,730
Invisible UHP	–	DupIP	1,609	1,531	686	296	4,122
Total			18,802	5,294	6,584	1,587	195,525

Table 7.2: Raw number of tunnels discovered by TNT according to their type. "Mix" refers to tunnels partially discovered with DPR, and partially with BRPR. No additional revelation technique is necessary for explicit and implicit tunnels.

opaque tunnels, as their content could not be exposed in 98% of the cases.

The proportion of invisible tunnels is not negligible (16% of the tunnels in the dataset). More precisely, PHP is the most prominent configuration, and is performed 87% of the time. This observation confirms the results of the survey presented in the previous chapter (see Section 6.1). RTL is able to infer the presence of more than half of the PHP invisible tunnels. A partial reason is the order in which TNT evaluates the triggers. Indeed, as indicated in Listing 7.3, the tool considers RTL before RFPLA, as it is proven to be more reliable. Consequently, in case of a successful revelation, the algorithm concludes that the the tunnel was identified based on RTL, even if RFPLA was also positive. DPR works better than BRPR in case of PHP, which is expected, as the technique is able to reveal LSPs ending with a JUNIPER device in its default configuration, and therefore identified with RTL. It is worth noting that with UHP, a buddy IP address was required in nearly 25% of the cases. In the other situations, applying directly BRPR or DPR was enough to expose the tunnel content. Consequently, UHP seems often deployed for transit traffic only. For example, at the egress LERs, the Explicit NULL Label is announced for the loopback addresses, while the other prefixes are advertised with the Implicit NULL label (i.e., PHP, and therefore BRPR works), or are not injected in LDP at all (DPR works).

The column labeled "mix" corresponds to tunnels partially revealed with BRPR, and partially with DPR. Typically, this pattern comes from *heterogeneous MPLS clouds*. For instance, operators may deploy both JUNIPER and CISCO hardware in their network without any homogeneous prefix distribution. Indeed, as already stated previously (see Section 7.2), by default, LDP only announces loopback addresses in JUNIPER devices, while it maps a label for each prefix in the IGP routing table in CISCO routers. Note that UHP and PHP label popping techniques may co-exist in an MPLS domain. Although not explained in Section 7.3 for clarity reasons, TNT can deal with those more complex architectures, making the tool quite robust to the pitfalls encountered on the Internet (5% of the invisible tunnels that were observed).

The column labeled “1Hop-LSP” corresponds to tunnels composed of a single internal LSR, where DPR and BRPR cannot be distinguished. Their large proportion (20%) is coherent with Figures 4.12 and 6.8, as well as with previous works [53, 137] that already signaled the existence of a high number of short explicit MPLS tunnels.

It is worth mentioning that the different results on invisible tunnels clearly contradict the conclusions of Donnet et al. [53]. Indeed, the authors suggested that they were probably 40 to 50 times less numerous than explicit ones, which is not the case in practice.

Figure 7.10 presents the distribution of MPLS tunnels according to their length. In 60% of the cases, LSPs contain less than 4 internal LSRs, all types of tunnel combined. They are thus rather short, which is not surprising in practice, because MPLS is a technology used in transit networks, where packets are forwarded to an exit as fast as possible in order to reduce resource consumption (hot-potato routing). This observation also confirms previous results on invisible (Figure 6.8) and explicit tunnels (Figure 4.12). In addition, opaque and implicit LSPs seem shorter than explicit ones, while invisible tunnels are a little bit longer. Finally, note that `TNT` may have been unable to reveal more than one hop for some UHP invisible tunnels. As this hop corresponds to the egress LER, the tunnel length is equal to 0 (no internal LSR was exposed). It explains why the grey curve (“UHP Invisible”) starts with a Y value different from 0.

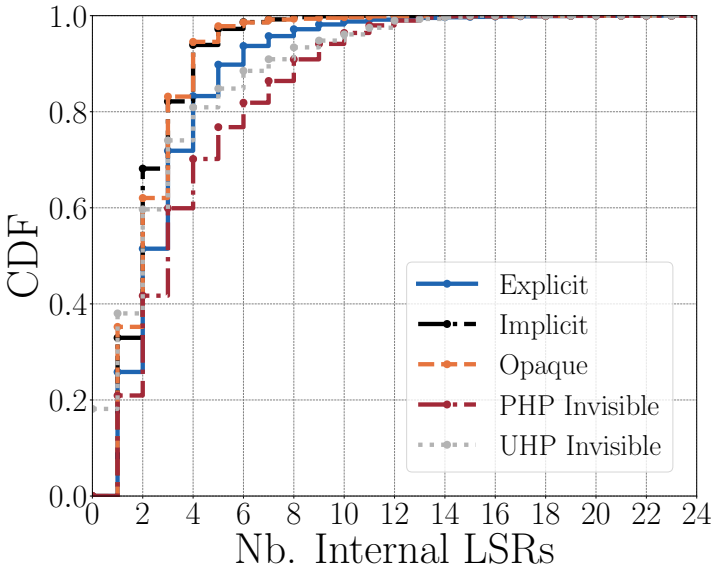


Figure 7.10: Tunnel length distribution.

Finally, Figure 7.11 illustrates the distance of MPLS tunnels to the monitor from which they were identified. This distance is defined as the position of the ingress LER in the trace between the source and the destination. At least 80% of the tunnels start at least four IP hops away from their monitor, while less than 20% appear after the 16th hop. Therefore, they are located, most of the time, outside the ISP of their measurement point. We can conclude that MPLS is more

prevalent in the core of the Internet (i.e., in Tier-1 ASs) than at the edge (i.e., in Stub ASs). The figure also shows that opaque and implicit tunnels seem to be found further apart from their monitor than other types of tunnels. Note that invisible LSPs appear generally closer to the measurement points due to a limitation of `TNT`. Indeed, as already mentioned in Section 7.3.3, the tool does not look for consecutive invisible tunnels in a trace. Only the closest to the monitor is revealed.

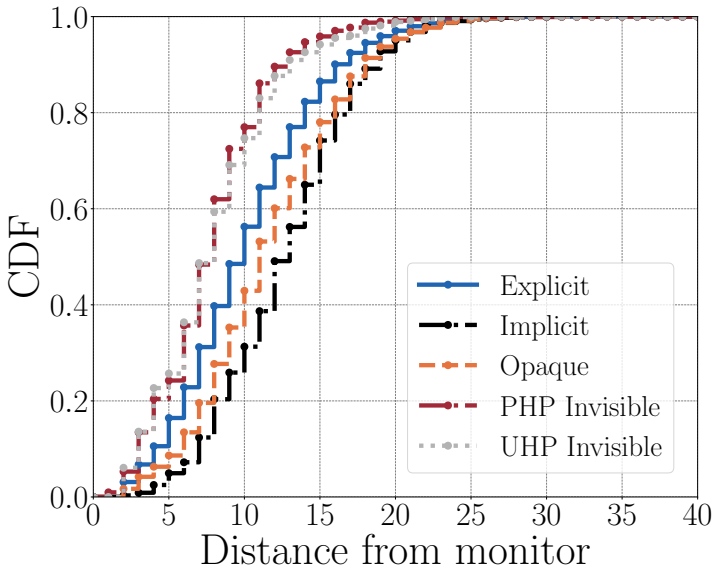


Figure 7.11: Distance of tunnels to their measurement point.

7.6 Conclusion

This chapter introduced `TNT` (*Trace the Naughty Tunnels*), an extension to `paris-traceroute` for revealing all MPLS tunnels along a path. The tool works in two simple steps. First, it runs `traceroute` towards a destination, and searches in the output any evidence of the presence of visible MPLS tunnels. Then, based on triggers, it estimates if an invisible LSP may exist between two hops in the trace, and performs additional measurements in order to expose its content, if any. The different inference and revelation techniques used by `TNT` were validated based on GNS3 simulations. The tool was also calibrated properly thanks to a *Receiver Operating Characteristic* curve obtained by varying its parameters in multiple measurement campaigns. Finally, in order to demonstrate its usefulness, `TNT` was run on multiple Archipelago monitors to collect data that was used to update the current view of MPLS deployment over the Internet.

It is worth signaling that `TNT` has been developed with a reproducibility perspective. Consequently, its source code, as well as the datasets and scripts used to generate the different results presented in this chapter, are publicly available [55]

CONCLUSION

Today's Internet is a huge interconnection of networks managed independently of each other. Each operator may decide to deploy specific technologies to forward packets inside his network. One of these technologies is MPLS. Although the protocol is heavily used on the Internet, it was not really investigated by the research community prior to this thesis. From a measurement perspective, MPLS could therefore be studied in two different ways:

(i) Only two works focused on the identification of MPLS tunnels in `traceroute` data [53, 137]. Unfortunately, they lack of precision in some of their models or do not take into account all possible configurations of the protocol. Consequently, our knowledge on its deployment on the Internet was incomplete. In addition, its use in networks, as well as the architectural principles followed by operators were never evaluated.

(ii) Network operators may configure their MPLS tunnels in order to hide their content to `traceroute` exploration, leading so to incomplete Internet topology data. This incompleteness may cause artifacts when it comes to modeling the worldwide network. Indeed, the resulting graphs may contain errors, such as false IP links, or nodes with an artificially high degree. Consequently, inferred Internet characteristics, such as path length or node degree distributions, may be biased. Even if hidden routers were already investigated by a few researchers [102, 103, 135, 136], their revelation from `traceroute` measurements was still an open question.

This is exactly what was tackled in this thesis: improvement of the understanding and knowledge of the deployment and use of MPLS, and presentation of new techniques (implemented in an integrated tool) for revealing most routers hidden by MPLS.

More specifically, at the architectural level, a lightweight fingerprinting technique was first presented. It consists in sending multiple probes towards a target in order to solicit ICMP responses whose TTLs can be used to infer a signature of n initial TTL values. With only two ICMP replies obtained with `traceroute` and `ping`, valuable signatures can be derived. They allow to discriminate between major router platforms. Thanks to a measurement campaign, this technique highlighted that ASs deploy mostly CISCO and JUNIPER devices in their MPLS network. However, as some parts of the standard documentation (RFCs) are open to a number of interpretations, default MPLS behaviors are different for both vendors. For instance, while

JUNIPER equipments only inject loopback addresses in LDP to exclusively forward in-transit packets with MPLS, CISCO devices announce a label for each prefix in their IGP routing table. Another example is the default independent LSP control mode of LDP in CISCO routers that may generate a specific type of MPLS tunnels called opaque tunnels. The fingerprinting method finds a suitable application here: identifying router brands allows to discriminate between particular MPLS implementations, and can therefore be used to guide the tunnel revelation techniques.

In practice, MPLS has different fields of applications on the Internet, such as VPNs and fast reroute. In transit networks, it is mostly deployed to forward packets between the border routers of the domain. In this way, the internal architecture can be simplified. This thesis introduced LPR (*Label Pattern Recognition*), an algorithm able to evaluate the use of MPLS in transit networks based on a label analysis at LSRs that are common to LSPs established between the same pair of ingress and egress LERs. Applying the algorithm on a `traceroute` dataset showed that the use of the protocol varies greatly from an operator to another. Indeed, we demonstrated that some of them tune the performance of their network thanks to traffic engineering solutions, or balance the traffic load on multiple equal-cost paths, while the vast majority simply always use the same LSP between two border devices.

However, a study of MPLS in IPv6 networks demonstrated that it is used differently in IPv6 than in IPv4. Indeed, a 6PE-like architecture built based on stacks of two labels is predominant in IPv6-capable networks. This MPLS structure allows to forward IPv6 traffic in dual-stack core networks where IPv6 versions of some protocols are not available (e.g., LDPv6), or where already established IPv4 tunnels can be reused.

At the measurement level, this thesis introduced different techniques able to infer the presence of invisible MPLS tunnels, and reveal their content when possible. On the one hand, hidden LSPs are signaled by RFPLA (*Return/Forward Path Length Asymmetry*), that compares the lengths of forward and return paths, and RTL (*Return Tunnel Length*), that exploits the asymmetry of the routes followed by `time exceeded` and `echo reply` messages. On the other hand, invisible LSRs can be exposed by DPR (*Direct Path Revelation*) with a single additional `traceroute`, or by BRPR (*Backward-Recursive Path Revelation*) with a recursive probing approach. While DPR works primarily in networks that only inject the loopback addresses of the border routers in LDP (default implementation in JUNIPER equipment), BRPR is able to reveal tunnels in domains that use MPLS to reach all the IGP prefixes (default implementation in CISCO devices).

All these techniques were validated using GNS3 simulations, and cross-validated based on a large-scale measurement campaign. They were also integrated into *Trace the Naughty Tunnels (TNT)*, a tool able to identify all kinds of MPLS tunnel along a path towards a specific destination. `TNT` is publicly available [55], and can be used by the research community in multiple fields. For instance, it allows to update the current quantification of MPLS tunnels in the worldwide network. It may also be run in order to detect hidden MPLS routers, and improve network

topology models, such as the distributions of node degrees and Internet path lengths.

Finally, the last observation in this work is that, even if MPLS is widely deployed by network operators, all its functionalities and possible configurations are not always well understood. Consequently the protocol may be the source of unexpected behaviors that are difficult to identify and interpret in practice. This situation can especially arise in heterogeneous networks deploying both CISCO and JUNIPER devices in their default mode, without unifying their configuration at the network scale.

Future Work and Research Directions

This thesis presented a first detailed characterization of MPLS in the Internet. Even if the protocol was studied in different ways, several aspects and improvements can still be investigated in future works.

More specifically, at the architectural level, one could envision to extend the basic signature of the fingerprinting method. Adding new fields should enlarge the spectrum of possible classes, making them more discriminant. In this way, new types of routers and manufacturers could possibly be identified, and perhaps, new revelation techniques for invisible MPLS tunnels could be designed. In addition, a new multi-probing `traceroute` tool could be developed to infer equipment-based paths on the Internet, and in particular in IP and MPLS domains. Generally speaking, it could be used to understand whether IP and MPLS networks are heterogeneous in terms of hardware and software, and to analyze a new OS deployment, or the market share evolution at different scales. However, in order to achieve good performance, the probing overhead should be kept as low as possible while completing the n-tuple with additional and possibly orthogonal features. Examples of information that could be taken into account are the initial TTL of other types of ICMP messages, the packet size, the LSE-TTL field, and the MPLS label range.

The fingerprinting technique could also be applied to IPv6 in order to check if routers initialize the IPv4 TTL and the IPv6 Hop Limit with the same value, and therefore, if new signatures could be derived from initial Hop Limit values.

At a more general level, the fingerprinting technique could be used in order to improve network measurement techniques. For instance, alias resolution methods could be speeded up. Indeed, if two IP addresses are aliases, they belong to the same router, and must therefore have the same signature. In this way, two addresses having different signatures do not belong to the same device, and should not be evaluated by alias resolution mechanisms [72].

In addition, the fingerprinting technique could also find a suitable use in network security. Indeed, if a specific routing equipment is known to have security breaches or vulnerabilities, knowing its signature may allow to identify it easily in a network, and determine possible points of failure, as well as weaknesses in the infrastructure.

The ability of fingerprinting routing devices based on network measurements offers a lot of

other possibilities to the research community. For instance, a study may discover that the routers associated to a given manufacturer or operating system behave differently than other devices in specific situations (higher treatment delay, ICMP rate limitation [124], forwarding anomalies, etc.). Researchers may also use the fingerprinting mechanism in order to limit an analysis to a specific type of routers.

Then, the use of MPLS in the Internet was studied based on the *Label Pattern Recognition (LPR)* algorithm. This algorithm can be validated and improved in different directions in order to extend its scope and increase its accuracy. First, LPR makes the assumption that a network enabling traffic engineering performs traffic differentiation according to destination prefixes. However, in practice, some operators may consider other ways for distinguishing types of service, such as, for example, the source IP prefix, the incoming AS, or even other fields from both IP and transport headers. These possible scenarios could be taken into account in order to optimize the algorithm. However, it is worth mentioning that basic `traceroute` measurements do not bring enough information for such a specific study. A dedicated and finely-calibrated probing campaign would therefore be necessary to better understand whether such practices are common or not.

Second, `paris-traceroute` and its *Multipath Detection Algorithm (MDA)* extension [11] could be run in order to validate the label-based technique allowing to differentiate LDP from RSVP-TE. Indeed, an extensive measurement campaign would allow to check that the LSPs tagged as Mono-FEC ECMP (resulting from LDP) by LPR are also visible with `paris-traceroute`, and that the ones classified as Multi-FEC LSPs (resulting from RSVP-TE) are not. If these properties are correct, they could constitute a ground proof of LPR. Moreover, such a validation campaign may also help to determine if the IP-only load balancers (i.e., not using TCP or UDP ports) identified by Augustin et al. [11] actually exist, or correspond to ingress LERs attached to multiple LSPs deployed on a per-destination basis.

Third, the current version of LPR does not rely on alias resolution mechanisms to avoid well-known biases that they may induce. However, defining an IOTP (*In-Out Transit Pair*) at the router level rather than at the IP level should provide more refined data for distinguishing the different classes. Indeed, it should reduce the number of IOTPs, and so, provide more consistent results that may be closer to the actual use of MPLS.

Last, LPR could be modified to work on LSP trees instead of IOTPs. In this way, it could take into account all the LSPs starting from several ingress LERs, and arriving at the same egress LER. This technique could bring a better understanding of the underlying use of LDP, and so, probably improve the relevance of the classification. Indeed, more LSPs should be grouped together and classified by LPR because they would only be indexed based on the egress LER. Note that, in practice, the use of ECMP would force the analysis of directed acyclic graphs instead of trees.

Apart from improving LPR, the techniques implemented in the algorithm may also be used in order to improve alias resolution mechanisms. Indeed, we have seen that explicit MPLS tunnels

are largely deployed by network operators. Therefore, if MPLS labels can be associated to two different IP addresses that appear between the same pair of ingress and egress LERs, these addresses are probably aliases if the label values observed for each of them are identical (principle of parallel links, as explained in Section 4.3.2).

One could also use LPR to study how network operators perform *Traffic Engineering (TE)* in their networks. Based on the collected data, the performance and architecture of existing TE solutions could be evaluated, and new algorithms could be suggested in order to improve current TE schemes [31].

At the measurement level, TNT is only able to reveal the first invisible tunnel in a trace. This limitation is due to the shadow effect that applies to RFPLA and RTL after a tunnel, as mentioned in Section 7.3.3. A possible improvement of TNT would be to detect this shadow effect, and configure the tool to not take into account the triggers while it applies. In this way, TNT would be able to discover more than one hidden LSP per destination. Note however that this technique does not allow to detect an invisible tunnel whose egress LER is located in the shadow of another one.

In addition, four distinct router OSs were considered in GNS3 to validate TNT. One could envision to extend this validation with more router manufacturers and OSs. In this way, new specific MPLS behaviors might be identified, and taken into account in future updates of TNT.

As mentioned in Table 7.2, the revelation techniques do not perform well with opaque tunnels. Carrier-of-carriers VPNs[1] may explain this result. A recent emulation in GNS3 performed by J.-R. Luttringer showed that this kind of architecture may generate opaque tunnels. Indeed, in the case of a VPN, stacks of 2 labels are used in the LSP, as mentioned in Section 1.6.1. If PHP is enabled, the PH pops the top label. However, as the bottom label represents the VPN route, its value is different from the Explicit and Implicit NULL labels, the standard end-of-tunnel labels. Consequently, the LSP ends abruptly on the exit border router, and an opaque tunnel appears. However, in this specific situation, CISCO egress LERs do not reply to `traceroute` probes with their incoming interface (i.e., the one that received the probe), but with the interface that would have been used to reach the customer VPN. As this address is not internal to the MPLS cloud, DPR and BRPR do not work. Even if carrier-of-carriers VPNs may explain the bad results of DPR and BRPR on opaque tunnels, the simulation performed in GNS3 was only a preliminary study. It should be deepened, and different tests should be driven in real networks in order to confirm this assumption, and maybe improve the revelation methods according to the findings.

One could also envision to run TNT in IPv6 networks to identify all kinds of MPLS tunnels. In this way, the state-of-the-art view of MPLS deployment in IPv6 could be updated thanks to a new detailed quantification taking into account possible invisible LSPs.

It is worth noting that TNT was developed during an internship at CAIDA in San Diego. At the time of writing, the tool is deployed on 35 Archipelago monitors, but is not in production

[1] A carrier-of-carriers VPN is observed when a “MPLS VPN-based service provider allows other service providers to use a segment of its backbone network" [35].

mode. In a near future, TNT should be run over the whole set of measurement points, and collect data that will be publicly available for the research community in order to allow numerous new studies. For instance, the collected traces containing the revealed MPLS devices may be analyzed in order to correct Internet models, such as the average IP path length or the distribution of the node degrees, and justify traceroute anomalies, such as abnormal delays or duplicate IP addresses observed due to invisible tunnels. Other researchers may use the data to infer the complete topology of a network, study its internal architecture, and suggest improvements to the existing infrastructure.

Finally, as TNT is publicly available, network operators may use it in order to discover if parts of their internal architecture can be exposed, see if the revealed information may lead to security issues, and find other solutions to hide their internal devices to traceroute explorations.

BIBLIOGRAPHY

[1] R. A. STEENBERGEN, *A Practical Guide to (Correctly) Troubleshooting with Traceroute*.
https://www.nanog.org/meetings/nanog47/presentations/Sunday/RAS_Traceroute_N47_Sun.pdf.
Talk given during the 47th Nanog meeting.

[2] ——, *MPLS for Dummies*.
https://www.nanog.org/meetings/nanog49/presentations/Sunday/mpls-nanog49.pdf.
Talk given during the 49th Nanog meeting.

[3] P. AGARWAL AND B. AKYOL, *Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks*, RFC 3443, Internet Engineering Task Force, January 2003.

[4] Z. AL-QUDAH, M. ALSARAYREH, I. JOMHAWY, AND M. RABINOVICH, *Internet Path Stability: Exploring the Impact of MPLS Deployment*, in Proc. IEEE Global Communications Conference (GLOBECOM), December 2016.

[5] AMERICAN REGISTRY FOR INTERNET NUMBERS (ARIN), *IPv4 Depletion*.
https://www.arin.net/resources/request/ipv4_countdown.html, September 2015.
Accessed: June 5, 2018.

[6] L. ANDERSSON AND R. ASATI, *Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field*, RFC 5462, Internet Engineering Task Force, February 2009.

[7] L. ANDERSSON, I. MINEI, AND B. THOMAS, *LDP Specification*, RFC 5036, Internet Engineering Task Force, October 2007.

[8] ARIN, *American Registry for Internet Numbers*.
https://www.arin.net/.
Accessed: June 5, 2018.

[9] R. ASATI, C. PIGNATARO, K. RAZA, V. MANRAL, AND R. PAPNEJA, *Updates to LDP for IPv6*, RFC 7552, Internet Engineering Task Force, June 2015.

[10] B. AUGUSTIN, X. CUVELLIER, B. ORGOGOZO, F. VIGER, T. FRIEDMAN, M. LATAPY, C. MAGNIEN, AND R. TEIXEIRA, *Avoiding Traceroute Anomalies with Paris Traceroute*, in Proc. ACM Internet Measurement Conference (IMC), October 2006.

[11] B. AUGUSTIN, R. TEIXEIRA, AND T. FRIEDMAN, *Measuring Load-Balanced Paths in the Internet*, in Proc. ACM Internet Measurement Conference (IMC), November 2007.

[12] D. AWDUCHE, L. BERGER, D. GAN, T. LI, V. SRINIVASAN, AND G. SWALLOW, *RSVP-TE: Extensions to RSVP for LSP Tunnels*, RFC 3209, Internet Engineering Task Force, December 2001.

[13] D. AWDUCHE, A. CHIU, A. ELWALID, I. WIDJAJA, AND X. XIAO, *Overview and Principles of Internet Traffic Engineering*, RFC 3272, Internet Engineering Task Force, May 2002.

[14] D. AWDUCHE, A. HANNAN, AND X. XIAO, *Applicability Statement for Extensions to RSVP for LSP-Tunnels*, RFC 3210, Internet Engineering Task Force, December 2001.

[15] D. AWDUCHE, J. MALCOLM, J. AGOGBUA, M. O'DELL, AND M. J., *Requirements for Traffic Engineering Over MPLS*, RFC 2702, Internet Engineering Task Force, September 1999.

[16] D. AYDIN, *CISCO vs. Juniper MPLS*.
`http://monsterdark.com/cisco-vs-juniper-mpls/`, June 2014.
Accessed: June 12, 2018.

[17] Y. BAR-YAM, *Concepts: Power Law*.
`http://www.necsi.edu/guide/concepts/powerlaw.html`, 2011.
Accessed: July 13, 2018.

[18] T. BATES, R. CHANDRA, D. KATZ, AND Y. REKHTER, *Multiprotocol Extensions for BGP-4*, RFC 4760, Internet Engineering Task Force, January 2007.

[19] R. BEVERLY, *Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery*, in Proc. ACM Internet Measurement Conference (IMC), November 2016.

[20] R. BEVERLY, A. BERGER, AND G. XIE, *Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization*, in Proc. ACM Internet Measurement Conference (IMC), November 2010.

[21] R. BONICA, D. GAN, D. TAPPAN, AND C. PIGNATARO, *ICMP Extensions for Multiprotocol Label Switching*, RFC 4950, Internet Engineering Task Force, August 2007.

[22] C. J. BOVY, H. T. MERTODIMEDJO, G. HOOGHIEMSTRA, H. UIJTERWAAL, AND P. VAN MIEGHEM, *Analysis of End-to-end Delay Measurements in Internet*, in Proc. Passive and Active Measurement Conference (PAM), March 2002.

[23] R. BRADEN, L. ZHANG, S. BERSON, S. HERZOG, AND S. JAMIN, *Resource ReSerVation Protocol (RSVP)*, RFC 2205, Internet Engineering Task Force, September 1997.

[24] V. BREÑA-MEDINA, *University of Bristol Thesis Template*.
https://fr.overleaf.com/latex/templates/university-of-bristol-thesis-template/kzqrfvyxxcdm/.
Accessed: April 12, 2018.

[25] B. CARPENTER AND S. BRIM, *Middleboxes: Taxonomy and Issues*, RFC 3234, Internet Engineering Task Force, February 2002.

[26] J. CASE, M. FEDOR, M. SCHOFFSTALL, AND J. DAVIN, *OSI IS-IS Intra-domain Routing Protocol*, RFC 1157, Internet Engineering Task Force, May 1990.

[27] I. CASTRO, J. C. CARDONA, S. GORINSKY, AND P. FRANCOIS, *Remote Peering: More Peering without Internet Flattening*, in Proc. ACM CoNEXT, December 2014.

[28] CENTER FOR APPLIED DATA ANALYSIS, *AS Rank*.
http://as-rank.caida.org/.
Accessed: June 7, 2018.

[29] ——, *Macroscopic Internet Topology Data Kit (ITDK)*.
http://www.caida.org/data/internet-topology-data-kit, March 2016.
Accessed: June 11, 2018.

[30] CENTER FOR APPLIED INTERNET DATA ANALYSIS (CAIDA), *AS Core: Visualizing IPv4 Internet Topology at a Macroscopic Scale in 2017*.
https://www.caida.org/research/topology/as_core_network/2017/.
Accessed: July 5, 2018.

[31] M. CHIESA, G. RÉTVÁRI, AND M. SCHAPIRA, *Oblivious Routing in IP Networks*, IEEE/ACM Transactions on Networking, vol. 26, no. 3, pp. 1292–1305, June 2018.

[32] B. CHUN, D. CULLER, T. ROSCOE, A. BAVIER, L. PETERSON, M. WAWRZONIAK, AND M. BOWMAN, *Planetlab: An Overlay Testbed for Broad-coverage Services*, ACM SIGCOMM Computer Communication Review, vol. 33, no. 3, pp. 3–12, July 2003.

[33] CISCO SYSTEMS, *Best Practices in Core Network Capacity Planning*.
https://www.cisco.com/c/en/us/products/collateral/routers/wan-automation-engine/white_paper_c11-728551.pdf.
Accessed: July 11, 2018.

[34] ——, *Cisco IOS NetFlow*.

https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.
Accessed: July 11, 2018.

[35] ——, *MPLS VPN Carrier Supporting Carrier with BGP*.
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mp_ias_and_csc/configuration/15-mt/mp-ias-and-csc-15-mt-book/mp-carrier-bgp.pdf.
Accessed: July 18, 2018.

[36] ——, *6PE FAQ: Why Does 6PE Use Two MPLS Labels in the Data Plane?*
https://www.cisco.com/c/en/us/support/docs/multiprotocol-label-switching-mpls/mpls/116061-qa-6pe-00.html, April 2013.
Accessed: June 5, 2018.

[37] ——, *Segment Routing Global Block (SRGB)*.
http://www.segment-routing.net/tutorials/2016-09-27-segment-routing-global-block-srgb/, September 2016.
Accessed: May 16, 2018.

[38] K. CLAFFY, Y. HYUN, K. KEYS, M. FOMENKOV, AND D. KRIOUKOV, *Internet Mapping: from Art to Science*, in Proc. IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH), March 2009.

[39] A. CLAUSET AND C. MOORE, *Traceroute Sampling Makes Random Graphs Appear to Have Power Law Degree Distributions*, cond-mat 0312674, arXiv, February 2004.

[40] COMPUTER HISTORY MUSEUM, *Official Website*.
http://www.computerhistory.org/.
Accessed: July 4, 2018.

[41] CREATIVE COMMONS, *Attribution 4.0 International (CC BY 4.0)*.
https://creativecommons.org/licenses/by/4.0/deed.en.
Accessed: April 12, 2018.

[42] X. CUVELLIER, *paris-traceroute(8) - Linux User's Manual*.
http://manpages.ubuntu.com/manpages/trusty/man8/paris-traceroute.8.html.
Accessed: May 09, 2018.

[43] J. CZYZ, M. ALLMAN, J. ZHANG, S. IEKEL-JOHNSON, E. OSTERWEIL, AND M. BAILEY, *Measuring IPv6 Adoption*, in Proc. ACM SIGCOMM, August 2014.

[44] G. DAVILA REVELO, M. A. RICCI, B. DONNET, AND J. I. ALVAREZ-HAMELIN, *Unveiling the MPLS Structure on Internet Topology*, in Proc. Traffic Monitoring and Analysis Workshop (TMA), April 2016.

[45] N. DAVIS, *Initial TTL Values*.
`http://noahdavids.org/self_published/TTL_values.html`, November 2011.
Accessed: May 17, 2018.

[46] J. DE CLERCQ, D. OOMS, M. CARUGI, AND F. LE FAUCHEUR, *BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*, RFC 4659, Internet Engineering Task Force, September 2006.

[47] J. DE CLERCQ, D. OOMS, S. PREVOST, AND F. LE FAUCHEUR, *Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)*, RFC 4798, Internet Engineering Task Force, February 2007.

[48] L. DE GHEIN, *MPLS Fundamental: A Comprehensive Introduction to MPLS (Theory and Practice)*, CISCO Press, November 2006.

[49] S. DEERING AND R. HINDEN, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 8200, Internet Engineering Task Force, July 2017.

[50] G. DETAL, B. HESMANS, O. BONAVENTURE, Y. VANAUBEL, AND B. DONNET, *Revealing Middlebox Interference with Tracebox*, in Proc. ACM Internet Measurement Conference (IMC), October 2013.

[51] A. DHAMDHERE, M. LUCKIE, B. HUFFAKER, K. CLAFFY, A. ELMOKASHFI, AND E. ABEN, *Measuring the Deployment of IPv6: Topology, Routing, and Performance*, in Proc. ACM Internet Measurement Conference (IMC), November 2012.

[52] B. DONNET AND T. FRIEDMAN, *Internet Topology Discovery: a Survey*, IEEE Communications Surveys and Tutorials, vol. 9, no. 4, pp. 2–15, December 2007.

[53] B. DONNET, M. LUCKIE, P. MÉRINDOL, AND J.-J. PANSIOT, *Revealing MPLS Tunnels Obscured from Traceroute*, ACM SIGCOMM Computer Communication Review, vol. 42, no. 2, pp. 87–93, April 2012.

[54] B. DONNET, P. RAOULT, T. FRIEDMAN, AND M. CROVELLA, *Efficient Algorithms for Large-Scale Topology Discovery*, in Proc. ACM SIGMETRICS, June 2005.

[55] B. DONNET, Y. VANAUBEL, P. MÉRINDOL, AND J.-J. PANSIOT, *MPLS Tunnels Tracking*.
`http://www.montefiore.ulg.ac.be/~bdonnet/mpls/`.
Accessed: June 25, 2018.

[56] K. EDELINE AND B. DONNET, *A first look at the prevalence and persistence of middleboxes in the wild*, in Proc. International Teletraffic Congress (ITC), September 2017.

[57] B. EDGEWORTH, A. FOSS, AND R. GARZA RIOS, *IP routing on Cisco IOS, IOS XE, and IOS XR: How a router works*.
`http://www.ciscopress.com/articles/article.asp?p=2272154&seqNum=3`, January 2015.
Accessed: April 20, 2018.

[58] EDUPERT, *Original Van Jacobson / Unix / LBL Traceroute*.
`https://kb.pert.geant.net/PERTKB/VanJacobsonTraceroute`.
Accessed: July 3, 2018.

[59] M. ENGIN TOZAL AND K. SARAC, *TraceNET: an Internet Topology Data Collector*, in Proc. ACM Internet Measurement Conference (IMC), November 2010.

[60] P. ERDÖS AND A. RÉNYI, *On the Evolution of Random Graphs*, Publ. Math. Inst. Hung. Acad. Sci., vol. 5, pp. 17–61, 1960.

[61] M. FALOUTSOS, P. FALOUTSOS, AND C. FALOUTSOS, *On Power-Law Relationships of the Internet Topology*, in Proc. ACM SIGCOMM, September 1999.

[62] N. FEAMSTER, R. JOHARI, AND H. BALAKRISHNAN, *Implications of Autonomy for the Expressiveness of Policy Routing*, in Proc. ACM SIGCOMM, August 2005.

[63] C. FILSFILS, S. PREVIDI, L. GINSBERG, B. DECRAENE, S. LITKOWSKI, AND R. SHAKIR, *Segment Routing Architecture*, RFC 8402, Internet Engineering Task Force, June 2018.

[64] T. FLACH, E. KATZ-BASSETT, AND R. GOVINDAN, *Quantifying Violations of Destination-Based Forwarding on the Internet*, in Proc. ACM Internet Measurement Conference (IMC), November 2012.

[65] R. FONTUGNE, E. ABEN, C. PELSSER, AND R. BUSH, *Pinpointing Delay and Forwarding Anomalies Using Large-Scale Traceroute Measurements*, in Proc. ACM Internet Measurement Conference (IMC), November 2017.

[66] P. FRANCOIS, C. FILSFILS, J. EVANS, AND O. BONAVENTURE, *Achieving sub-second IGP convergence in large IP networks*, ACM SIGCOMM Computer Communication Review, vol. 35, no. 3, pp. 33–44, July 2005.

[67] W. GEORGE AND C. PIGNATARO, *Gap Analysis for Operating IPv6-Only MPLS Networks*, RFC 7439, Internet Engineering Task Force, January 2015.

[68] G. GESHEV, *Warranty Void if Label Removed: Attacking MPLS Networks*, in Proc. Zero Nights, December 2015.

[69] V. GIOTSAS, M. LUCKIE, B. HUFFAKER, AND K. CLAFFY, *IPv6 AS Relationships, Clique, and Congruence*, in Proc. Passive and Active Measurement Conference (PAM), March 2015.

[70] GNS3 TECHNOLOGIES INC., *GNS3*.
`https://www.gns3.com/`.
Accessed: June 12, 2018.

[71] A. GORIELY, *Applied Mathematics: A Very Short Introduction*, Oxford University Press, 2018.

[72] J.-F. GRAILET AND B. DONNET, *Towards a Renewed Alias Resolution with Space Search Reduction and IP Fingerprinting*, in Proc. Network Traffic Measurement and Analysis Conference (TMA), June 2017.

[73] T. G. GRIFFIN, F. B. SHEPHERD, AND G. WILFONG, *The Stable Paths Problem and Interdomain Routing*, IEEE/ACM Transactions on Networking, vol. 10, no. 2, pp. 232–243, April 2002.

[74] J.-L. GUILLAUME, M. LATAPY, AND C. MAGNIEN, *Comparison of Failures and Attacks on Random and Scale-Free Networks*, in Proc. 8th International Conference on Principles of Distributed Systems (OPODIS), December 2004.

[75] M. H. GUNES AND K. SARAC, *Resolving Anonymous Routers in the Internet Topology Measurement Studies*, in Proc. IEEE INFOCOM, April 2008.

[76] H. HADDADI, G. IANNACCONE, A. MOORE, R. MORTIER, AND M. RIO, *Network Topologies: Inference, Modeling and Generation*, IEEE Communications Surveys and Tutorials, vol. 10, no. 2, pp. 48–69, April 2008.

[77] J. HAWKINSON AND T. BATES, *Guidelines for creation, selection, and registration of an Autonomous System (AS)*, RFC 1930, Internet Engineering Task Force, March 1996.

[78] U. HENGARTNER, S. MOON, R. MORTIER, AND C. DIOT, *Detection and Analysis of Routing Loops in Packet Traces*, in Proc. ACM SIGCOMM Internet Measurement Workshop (IMW), November 2002.

[79] R. HINDEN AND S. DEERING, *IP Version 6 Addressing Architecture*, RFC 4291, Internet Engineering Task Force, February 2006.

[80] F. HOEBRECK, *Master Thesis : Improving Network Fingerprinting*.
`https://matheo.uliege.be/handle/2268.2/2524`, June 2017.
Accessed: July 12, 2018.

[81] IANA, *Internet Assigned Numbers Authority*.
`https://www.iana.org`.
Accessed: April 20, 2018.

[82] V. JACOBSON, *Pathchar - a Tool to Infer Characteristics of Internet Paths*.
`ftp://ftp.ee.lbl.gov/pathchar/msri-talk.pdf`, April 1997.
Accessed: July 12, 2018.

[83] L. JACQUIN, V. ROCA, M. A. KAAFAR, F. SCHULER, AND J. L. ROCH, *IBTrack: An ICMP Black holes Tracker*, in Proc. IEEE Global Communications Conference (GLOBECOM), December 2012.

[84] D. KATZ, *IP Router Alert Option*, RFC 2113, Internet Engineering Task Force, February 1997.

[85] D. KATZ, K. KOMPELLA, AND D. YEUNG, *Traffic Engineering (TE) Extensions to OSPF Version 2*, RFC 3630, Internet Engineering Task Force, September 2003.

[86] E. KATZ-BASSETT, H. MADHYASTHA, V. ADHIKARI, C. SCOTT, J. SHERRY, P. VAN WESEP, A. KRISHNAMURTHY, AND T. ANDERSON, *Reverse Traceroute*, in Proc. USENIX Symposium on Networked Systems Design and Implementations (NSDI), June 2010.

[87] M. H. KEIO, Y. NISHIDA, C. RAICIU, A. GREENHALGH, M. HANDLEY, AND H. TOKUDA, *Is It Still Possible to Extend TCP*, in Proc. ACM Internet Measurement Conference (IMC), November 2011.

[88] K. KEYS, *Internet-Scale IP Alias Resolution Techniques*, ACM SIGCOMM Computer Communication Review, vol. 40, no. 1, pp. 50–55, January 2010.

[89] T. KOHNO, A. BROIDO, AND K. CLAFFY, *Remote Physical Device Fingerprinting*, IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 2, pp. 93–108, May 2005.

[90] K. KOMPELLA, J. DRAKE, S. AMANTE, W. HENDERICKX, AND L. YONG, *The Use of Entropy Labels in MPLS Forwarding*, RFC 6790, Internet Engineering Task Force, November 2012.

[91] K. KOMPELLA, M. HELLERS, AND R. SINGH, *Multi-Path Label Switched Paths Signaled Using RSVP-TE*, Internet Draft (Work in Progress) draft-kompella-mpls-rsvp-ecmp-06, Internet Engineering Task Force, March 2015.

[92] J. F. KUROSE AND K. W. ROSS, *Computer Networking: A Top-Down Approach*, Pearson, 6 ed., 2012.

[93] A. LAKHINA, J. BYERS, M. CROVELLA, AND P. XIE, *Sampling Biases in IP Topology Measurements*, in Proc. IEEE INFOCOM, April 2003.

[94] M. LEBER, *IPv6 Internet Broken, Cogent/Telia/Hurricane not Peering*.
`http://mailman.nanog.org/pipermail/nanog/2009-October/014017.html`, Nanog Mailing-list, October 2009.
Accessed: June 7, 2018.

[95] T. LI AND H. SMIT, *IS-IS Extensions for Traffic Engineering*, RFC 5305, Internet Engineering Task Force, October 2008.

[96] M. LUCKIE, *Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet*, in Proc. ACM Internet Measurement Conference (IMC), November 2010.

[97] M. LUCKIE, A. DHAMDHERE, B. HUFFAKER, D. CLARK, AND K. CLAFFY, *bdrmap: Inference of borders between IP networks*, in Proc. ACM Internet Measurement Conference (IMC), November 2016.

[98] M. LUCKIE, Y. HYUN, AND B. HUFFAKER, *Traceroute Probe Method and Forward IP Path Inference*, in Proc. ACM Internet Measurement Conference (IMC), October 2008.

[99] G. F. LYON, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*, Nmap Project, 2009.
`http://nmap.org/book/toc.html`.

[100] D. MAGONI AND J.-J. PANSIOT, *Analysis of the Autonomous System Network Topology*, ACM SIGCOMM Computer Communication Review, vol. 31, no. 3, pp. 26–37, July 2001.

[101] G. MALKIN, *RIP Version 2*, RFC 2453, Internet Engineering Task Force, November 1998.

[102] P. MARCHETTA, W. DE DONATO, V. PERSICO, AND A. PESCAPÉ, *Experimenting with Alternative Path Tracing Solutions*, in Proc. IEEE Symposium on Computers and Communications (ISCC), July 2015.

[103] P. MARCHETTA AND A. PESCAPÉ, *DRAGO: Detecting, Quantifying and Locating Hidden Routers in Traceroute IP Paths*, in Proc. Global Internet Symposium (GI), April 2013.

[104] J. S. MARCUS, *Designing Wide Area Networks and Internetworks: a Practical Guide*, Addison-Wesley, 1999.

[105] M. MATHIS AND J. MAHDAVI, *Diagnosing Internet Congestion with a Transport Layer Performance Tool*, in Proc. International Networking Conference (INET), June 1996.

[106] A. MEDINA, M. ALLMAN, AND S. FLOYD, *Measuring Interactions Between Transport Protocols and Middleboxes*, in Proc. ACM Internet Measurement Conference (IMC), October 2004.

[107] P. MÉRINDOL, B. DONNET, O. BONAVENTURE, AND J.-J. PANSIOT, *On the Impact of Layer-2 on Node Degree Distribution*, in Proc. ACM Internet Measurement Conference (IMC), November 2010.

[108] MERIT NETWORK, INC., *Official Website*.
`https://www.merit.edu/`.
Accessed: July 5, 2018.

[109] J. MOY, *OSPF Version 2*, RFC 2328, Internet Engineering Task Force, April 1998.

[110] K. MUTHUKRISHNAN AND A. MALIS, *A Core MPLS IP VPN Architecture*, RFC 2917, Internet Engineering Task Force, September 2000.

[111] M. MUUSS, *The Story of the PING Program*.
`http://ftp.arl.mil/mike/ping.html`.
Accessed: July 11, 2018.

[112] NATIONAL SCIENCE FOUNDATION, *A Brief History of NSF and the Internet*.
`https://www.nsf.gov/news/news_summ.jsp?cntn_id=103050`.
Accessed: July 5, 2018.

[113] NOCTION NETWORK INTELLIGENCE, *Tier 1 carrier performance: May, 2017 report*.
`https://www.noction.com/blog/tier-1-carrier-performance-may-2017-snapshot`, May 2017.
Accessed: May 07, 2018.

[114] D. ORAN, *OSI IS-IS Intra-domain Routing Protocol*, RFC 1142, Internet Engineering Task Force, February 1990.

[115] P. PAN, G. SWALLOW, AND A. ATLAS, *Fast Reroute Extensions to RSVP-TE for LSP Tunnels*, RFC 4090, Internet Engineering Task Force, May 2005.

[116] R. PASTOR-SATORRAS AND A. VESPIGNANI, *Evolution and Structure of the Internet: A Statistical Physics Approach*, Cambridge University Press, 2004.

[117] A. PATHAK, M. ZHANG, Y. C. HU, R. MAHAJAN, AND D. MALTZ, *Latency inflation with MPLS-based traffic engineering*, in Proc. ACM Internet Measurement Conference (IMC), November 2011.

[118] V. PAXSON, *End-to-End Routing Behavior in the Internet*, in Proc. ACM SIGCOMM, August 1996.

[119] J. POSTEL, *User Datagram Protocol*, RFC 768, Internet Engineering Task Force, August 1980.

[120] ——, *Internet Control Message Protocol*, RFC 792, Internet Engineering Task Force, September 1981.

[121] ——, *Internet Protocol*, RFC 791, Internet Engineering Task Force, September 1981.

[122] ——, *Transmission Control Protocol*, RFC 793, Internet Engineering Task Force, September 1981.

[123] ——, *Assigned Numbers*, RFC 1700, Internet Engineering Task Force, October 1994.

[124] R. RAVAIOLI, G. URVOY-KELLER, AND C. BARAKAT, *Characterizing ICMP Rate Limitation on Routers*, in Proc. IEEE International Conference on Communications (ICC), June 2015.

[125] REAL TIME STATISTICS PROJECT, *Internet Live Stats*.
`http://www.internetlivestats.com`.
Accessed: April 20, 2018.

[126] Y. REKHTER, T. LI, AND S. HARES, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, Internet Engineering Task Force, January 2006.

[127] J. REXFORD, *Network Protocols Designed for Optimizability*, in Proc. Annual Conference on Information Sciences and Systems (CISS), March 2006.

[128] E. ROSEN AND Y. REKHTER, *BGP/MPLS IP Virtual Private Networks (VPNs)*, RFC 4364, Internet Engineering Task Force, February 2006.

[129] E. ROSEN, D. TAPPAN, G. FEDORKOW, Y. REKHTER, D. FARINACCI, T. LI, AND A. CONTA, *MPLS Label Stack Encoding*, RFC 3032, Internet Engineering Task Force, January 2001.

[130] E. ROSEN, A. VISWANATHAN, AND R. CALLON, *Multiprotocol Label Switching Architecture*, RFC 3031, Internet Engineering Task Force, January 2001.

[131] M. ROUGHAN, M. THORUP, AND Y. ZHANG, *Traffic Engineering with Estimated Traffic Matrices*, in Proc. ACM Internet Measurement Conference (IMC), October 2003.

[132] A. SEBASTIAN, *Default Time To Live (TTL) Values*.
`http://www.binbert.com/blog/2009/12/default-time-to-live-ttl-values/`, December 2009.
Accessed: May 17, 2018.

[133] R. SEDGEWICK, *Algorithms in Java, Parts 1-4*, Addison-Wesley Professional, July 2002.

[134] S. SHENKER, C. PARTRIDGE, AND R. GUERIN, *Specification of Guaranteed Quality of Service*, RFC 2212, Internet Engineering Task Force, September 1997.

[135] R. SHERWOOD, A. BENDER, AND N. SPRING, *Discarte: a Disjunctive Internet Cartographer*, in Proc. ACM SIGCOMM, August 2008.

[136] R. SHERWOOD AND N. SPRING, *Touring the Internet in a TCP Sidecar*, in Proc. ACM Internet Measurement Conference (IMC), October 2006.

[137] J. SOMMERS, B. ERIKSSON, AND P. BARFORD, *On the Prevalence and Characteristics of MPLS Deployments in the Open Internet*, in Proc. ACM Internet Measurement Conference (IMC), November 2011.

[138] A. SOULE, H. RINGBERG, F. SILVEIRA, AND C. DIOT, *Challenging the Supremacy of Traffic Matrices in Anomaly Detection*, in Proc. ACM Internet Measurement Conference (IMC), October 2007.

[139] C. SRINIVASAN, L. P. BLOOMBERG, A. VISWANATHAN, AND T. NADEAU, *Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)*, RFC 3812, Internet Engineering Task Force, June 2004.

[140] TEAM CYMRU, *IP to ASN Mapping*.
`https://www.team-cymru.com/IP-ASN-mapping.html`.
Accessed: June 6, 2018.

[141] R. TEIXEIRA, K. MARZULLO, S. SAVAGE, AND G. M. VOELKER, *In Search of Path Diversity in ISP Networks*, in Proc. ACM Internet Measurement Conference (IMC), October 2003.

[142] UNIVERSITY OF OREGON, *University of Oregon Route Views Project*.

[143] Y. VANAUBEL, J.-R. LUTTRINGER, P. MÉRINDOL, J.-J. PANSIOT, AND B. DONNET, *TNT: Technical Report*, May 2018.
See `http://www.montefiore.ulg.ac.be/~bdonnet/mpls/Papers/mpls-techrep.pdf`.

[144] Y. VANAUBEL, P. MÉRINDOL, J.-J. PANSIOT, AND B. DONNET, *MPLS Under the Microscope: Revealing Actual Transit Path Diversity*, in Proc. ACM Internet Measurement Conference (IMC), October 2015.

[145] ——, *A Brief History of MPLS Usage in IPv6*, in Proc. Passive and Activement Measurement Conference (PAM), March/April 2016.

[146] ——, *Through the Wormhole: Tracking Invisible MPLS Tunnels*, in Proc. ACM Internet Measurement Conference (IMC), November 2017.

[147] Y. VANAUBEL, J.-J. PANSIOT, P. MÉRINDOL, AND B. DONNET, *Network Fingerprinting: TTL-Based Router Signature*, in Proc. ACM Internet Measurement Conference (IMC), October 2013.

[148] F. VEYSSET, O. COURTAY, AND O. HEEN, *New Tool and Technique for Remote Operating System Fingerprinting*.
`http://ouah.lescigales.org/ring-full-paper.pdf`, April 2002.
Accessed: May 17, 2018.

[149] S. VISSICCHIO, L. VANBEVER, C. PELSSER, L. CITTADINI, P. FRANCOIS, AND O. BONAVENTURE, *Improving Network Agility with Seamless BGP Reconfigurations*, IEEE/ACM Transactions on Networking, vol. 21, no. 3, pp. 990–1002, June 2013.

[150] F. WANG, Z. M. MAO, J. WANG, L. GAO, AND R. BUSH, *A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance*, in Proc. ACM SIGCOMM, August 2006.

[151] Y. WANG, M. SCHAPIRA, AND J. REXFORD, *Neighbor-Specific BGP: More Flexible Routing Policies While Improving Global Stability*, in Proc. ACM SIGMETRICS, June 2009.

[152] W. WILLINGER, D. ALDERSON, AND J. C. DOYLE, *Mathematics and the Internet: a Source of Enormous Confusion and Great Potential*, Notices of the American Mathematical Society, vol. 56, no. 5, pp. 586–599, May 2009.

[153] X. XIAO, A. HANNAN, AND B. BAILEY, *Traffic Engineering with MPLS in the Internet*, IEEE Network Magazine, vol. 14, no. 2, pp. 28–33, April 2000.