Let There Be Light: Revealing Hidden MPLS Tunnels with TNT

Jean-Romain Luttringer‡, Yves Vanaubel*, Pascal Mérindol‡, Jean-Jacques Pansiot‡, Benoit Donnet*
∗ Montefiore Institute, Université de Liège – Belgium
‡ Icube, Université de Strasbourg – France

***Abstract*—Internet topology discovery aims at analyzing one of the most complex distributed systems currently deployed. Usually, it relies on measurement campaigns using hop-limited probes sent with `traceroute`. However, this probing tool comes with several limits. In particular, some MPLS clouds might obfuscate collected traces. The resulting Internet maps, their inferred properties, and the graph models are thus incomplete and inaccurate.**

In this paper, we introduce `TNT` (Trace the Naughty Tunnels), an extension to Paris traceroute for revealing, or at least detect, all MPLS tunnels along a path. First, along with `traceroute` and `ping` probes, `TNT` looks for hints indicating the presence of hidden tunnels. Those hints are peculiar patterns in the resulting output, e.g., significant TTL shifts or duplicate IP addresses. Second, if those hints trigger alarms, `TNT` launches additional dedicated probing for possibly revealing hidden tunnels. We use GNS3 to reproduce, verify, and understand the limits and capabilities of `TNT` in a controlled environment. We also calibrate the thresholds at which alarms are triggered through a dedicated measurement campaign. Finally, we deploy `TNT` on the Archipelago platform and provide a quantified classification of MPLS configurations. All our results, including the data, the code, and the GNS3 experiments, are fully and publicly available.

***Index Terms*—`TNT`, MPLS, FRPLA, RTLA, BRPR, DPR, fingerprinting, taxonomy**

I. Introduction

For now twenty years, the Internet topology discovery has attracted a lot of attention from the research community [1, 2]. First, numerous tools have been proposed to better capture the Internet at the IP interface level (mainly based on `traceroute`) and at the router level (by aggregating IP interfaces of a router through *alias resolution*). Second, the data collected has been used to model the Internet [3], but also to have a better knowledge of the network ecosystem and how it is structured and organized by operators.

However, despite the work done so far, a lot of issues still remain, especially in data collection processes based on `traceroute`. For instance, collecting data about Layer-2 devices connecting routers is still an open question, although it has been addressed previously with a, nowadays, deprecated tool (i.e., IGMP-based probing [4]). Another example is the relationship between traditional network hardware and the so-called middleboxes [5, 6]. Last but not least, MPLS tunnels [7] also have an impact on topology discovery as they allow operators to hide internal hops, as highlighted by our previous works [8, 9].

This paper focuses on the interaction between `traceroute` and MPLS. In a nutshell, MPLS has been designed to reduce the time required to make forwarding decisions thanks to the insertion of *labels* (called *Label Stack Entries*, or LSE) before the IP header.[1] In an MPLS network, packets are forwarded using an exact match lookup of a 20-bit value found in the LSE. At each MPLS hop, the label of the incoming packet is swapped with its associated outgoing label (such a mapping being defined in a specific MPLS switching table). The MPLS forwarding engine is lighter than the IP one, as performing an exact match for a label is simpler than retrieving the longest matching prefix for an IP address.

Some MPLS tunnels may be visible to `traceroute` because MPLS routers are able to generate ICMP `time-exceeded` messages when the MPLS TTL expires. Since the ICMP message embeds the LSE, the presence of the tunnel is then obvious [8, 10]. However, MPLS supports optional features that make tunnels more or less invisible to `traceroute`. Such features modify the way routers process the IP and MPLS TTL of a packet. By carefully analyzing MPLS related patterns based on TTL values (e.g., the quoted TTL or the returned TTL of both error and standard replies), one can identify and possibly discover L3-hops hidden within an MPLS cloud. We already proposed a first attempt for revealing so-called Invisible tunnels [9].

This paper aims at improving the efficiency of this discovery in order to reveal (or at least identify) more invisible tunnels at a lower cost. This is achieved by introducing `TNT` (**T**race the **N**aughty **T**unnels), an open-source scamper [11] plugin extension based on Paris traceroute [12], that includes techniques for inferring, classifying, and possibly revealing MPLS tunnel content. In particular:

1) we strongly **revise the MPLS tunnel classification** we previously proposed [8]. In particular, we subdivide the "Invisible Tunnel" class in two more accurate categories, "Invisible PHP" and "Invisible UHP". We show that those tunnels can be systematically revealed when they are built with basic P2P LDP [13] or RSVP-TE [14] circuits, and can be at least detected if constructed with more complex technologies such as P2MP VPRN [15]. We also explain why the content of most "Opaque" tunnels cannot be revealed in practice. Finally, we refine and update the

[1] While MPLS can be used with IPv6, we only consider IPv4 in this paper.

previous quantification of each tunnel class with large-scale measurements performed in the wild;
2) we complement the state of the art with `traceroute`-based **measurement techniques** able to reveal most (or at least detect all) MPLS tunnels, even those built to hide their content. While our previous work [9] required to target suspect and pre-analyzed zones in the Internet (i.e., considering high degree nodes and their neighbors visible in the ITDK dataset [16]), `TNT` includes new fully integrated measurement techniques. We associate *indicators* or *triggers* with each category of tunnels within our classification. They are used to determine, on the fly, the potential presence of a tunnel and, possibly, its nature. In particular, in this paper, we are able to identify the presence of the newly introduced "UHP Invisible" tunnel class thanks to the duplication of an IP address in the `traceroute` output. When a trigger is pulled during a `traceroute` exploration, an MPLS *revelation* [9] is launched with the objective of revealing the tunnel content. We validate the indicators, triggers, and revelations using GNS3, an emulator running the actual OS of different brands of routers in a virtualized environment[2], on a large set of realistic configurations. We also show, through measurements, that our techniques are efficient in terms of cost (i.e., the additional amount of probes injected is reasonable, especially compared to the quality of new data discovered) and errors;
3) we **implement** those techniques within Scamper [11], the state of the art network measurements toolbox as a Paris traceroute extension, called `TNT`, and deploy it on the Archipelago infrastructure [17]. `TNT` aims at replacing the old version of Scamper and, as such, it is launched every day towards millions of destinations. We thus argue that `TNT` is useful to study MPLS deployment and usage over time, increasing so our knowledge and culture on this technology;
4) we **analyze** the data collected, the efficiency of `TNT` in doing so (for tuning it to its best set of calibration parameters) and report a new quantification on MPLS deployment in the wild, correcting and updating so previous results that erroneously underestimated or overestimated the prevalence of some tunnel classes [8]
5) we work in a **reproducibility** perspective. As such, all our code (`TNT`, GNS3 configurations and their experimental outcomes, data processing and analysis) as well as our collected dataset are made available.[3]

Compared to the conference version of this paper [18], we provide a more comprehensive description of MPLS mechanisms (Sec. II-B and II-C), a deeper discussion on `TNT` limits – in particular with respect to Opaque tunnels (Sec. V-B), a more refined calibration analysis (Sec. VI-C), and a deeper discussion on how `TNT` behaves in the wild (Table IV). We also provide additional explanation on the relationships between Opaque tunnels and VPRN, explaining so why Opaque tunnels content cannot be revealed (Fig. 6).

[2]See https://gns3.com/

[3]See http://www.montefiore.ulg.ac.be/~bdonnet/mpls

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 I: Summary of main router signature, the first initial TTL of the pair corresponds to ICMP `time-exceeded`, while the second is for ICMP `echo-reply`.

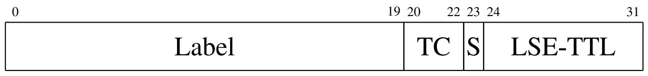


Fig. 1: The MPLS label stack entry (LSE) format.

The remainder of this paper is organized as follows: Sec. II provides the required technical background for this paper; Sec. III revises the MPLS taxonomy initially introduced by Donnet et al. [8] in the light of newly understood MPLS behaviors; Sec. IV describes our techniques for detecting and revealing hidden tunnels; Sec. V formally introduces `TNT`, our extension to `traceroute` for revealing the content of all MPLS tunnels; Sec. VI discusses `TNT` parameters and its calibration, while Sec. VII presents results of the `TNT` deployment over the Archipelago architecture; Sec. VIII positions our work with respect to the state of the art; finally, Sec. IX concludes this paper by summarizing its main achievements.

II. Background

This section discusses the technical background required for the paper. Sec. II-A explains how hardware brands can be inferred from collected TTLs. Sec. II-B provides the basics of MPLS labels and introduces the MPLS control plane. Eventually, Sec. II-C focuses on the MPLS data plane and the MPLS TTL processing. Table II provides a summary of the main acronyms related to the MPLS ecosystem and their corresponding concepts in the classic IP world. Moreover, Fig. 2 (upper part) illustrates the main vocabulary associated to MPLS tunnels.

A. Router Fingerprinting

Vanaubel et al. have presented in [19] a simple router fingerprinting technique that classifies networking devices according to their hardware and operating system (OS). This method infers the initial TTL values used by a router when forging different kinds of packets. It then builds a router *signature*, i.e., the n-tuple of n initial TTLs. A basic pair-signature (with $n = 2$) simply uses the initial TTL of two different messages: an ICMP `time-exceeded` message elicited by a `traceroute` probe, and an ICMP `echo-reply` message obtained from an `echo-request` probe. Table I summarizes the main router signatures, with associated router brands and router OSes. This feature is particularly useful in our study since the two most deployed router brands, Cisco and Juniper, have distinct MPLS behaviors and signatures that we exploit.

B. MPLS Basics and Control Plane Operations

MPLS routers, i.e., *Label Switching Routers* (LSRs), exchange labeled packets over *Label Switched Paths* (LSPs).

In practice, those packets are tagged with one or more *label stack entries* (LSE) inserted between the frame header (data-link layer) and the IP packet (network layer). Each LSE is made of four fields as illustrated by Fig. 1: an MPLS label used to forward the packet, a Traffic Class field (for quality of service, priority, and Explicit Congestion Notification [20]), a bottom of stack flag bit (to indicate whether the current LSE is the last in the stack [21])[4], and a time-to-live field (*LSE-TTL*) having the same purpose as the IP-TTL field [22] (i.e., avoiding routing loops).

Labels may be allocated through the *Label Distribution Protocol* (LDP) [13]. Each LSR announces to its neighbors the association between a prefix in its routing table and a label it has chosen for a given *Forwarding Equivalent Class* (a FEC is a destination prefix by default), populating so a *Label Forwarding Information Table* (LFIB) in each LSR. LDP is mainly used for scalability reasons (e.g., to limit BGP-IGP interactions to edge routers) and to avoid anomalies for the transit traffic such as iBGP deflection issues. Indeed, LDP deploys tunnels following the same routes as the IGP. Labels can also be distributed through RSVP-TE [14] when MPLS is used for Traffic Engineering (TE) purposes. In practice, most operators deploying RSVP-TE tunnels also use LDP [9] as an underlying default labeling protocol.

With LDP, MPLS has two ways of binding labels to destination prefixes: (i) through ordered LSP control (default configuration of Juniper routers [23]) and, (ii) through independent LSP control (default configuration of Cisco routers [24, Chap. 4]). In the former mode, an LSR only binds a label to a prefix if it is local (the LSR is the exit point of the LSP), or if it has received a label binding proposal from the IGP next hop towards this prefix. This mode is thus iterative as each intermediate upstream LSR waits for a proposal from its downstream LSR, building thus the LSP from the exit to the entry point. Juniper routers use this mode as default and only propose labels for loopback IP addresses.

In the second mode, the Cisco default one, an LSR creates a label binding for each prefix it has in its RIB, even if it is not directly connected to it. This label binding is then distributed to its neighbors. This mode does not require any proposal from downstream LSRs. Consequently, a label proposal is sent to all neighbors without ensuring that the LSP is enabled up to the wanted exit point. LSP setup takes less time but may lead to uncommon situations in which an LSP can end abruptly before the supposed exit point of the tunnel (see Sec. III for details).

The last LSR towards an FEC is the *Egress Label Edge Router* (the Egress LER – PE_2 in Fig. 2). Depending on its configuration, two labeling modes may be performed. The default mode [9] is *Penultimate Hop Popping* (PHP), where the Egress advertises an Implicit NULL label (label value of 3 [21]). In this case, the previous LSR (*Penultimate Hop LSR*, PH LSR – P_3 in Fig. 2) – is in charge of removing the LSE to reduce the load on the Egress. In the *Ultimate Hop Popping* (UHP) mode, the Egress LER advertises an Explicit NULL label (label value of 0 [21]). In this case, The PH LSR will swap the current label with an Explicit NULL label and the Egress LER will be responsible for its removal. Labels assigned by LSRs other than the Egress LER are distinct from Implicit or Explicit NULL labels. The *Ending Hop LSR* (EH) is the LSR in charge of removing the LSE, it can be the PH LSR in case of PHP, or the Egress LER in case of UHP[5].

Acronym	Meaning	IP
LSR	**L**abel **S**witching **R**outer	
PH LSR	**P**enultimate **H**op LSR	Router
EH	**E**nding **H**op LSR	
LER	**L**abel **E**dge **R**outer	Border Router
LSP	**L**abel **S**witching **P**ath	Tunnel
LSE	**L**abel **S**tack **E**ntry	Header
LSE-TTL	LSE **T**ime-**t**o-**L**ive	IP-TTL
LDP	**L**abel **D**istribution **P**rotocol	Signaling (control plane)
RSVP-TE	**R**e**S**er**V**etation **P**rotocol – **T**raffic **E**ngineering	
LIB	**L**abel **I**nformation **B**ase	RIB
LFIB	**L**abel **F**orwarding **I**nformation **B**ase	FIB
PHP	**P**enultimate **H**op **P**opping	Decapsulation
UHP	**U**ltimate **H**op **P**opping	
FEC	**F**orwarding **E**quivalent **C**lass	QoS Class

TABLE II: MPLS terminology with its classic IP matching.

C. MPLS Data Plane and TTL processing

Depending on its location along the LSP, an LSR applies one of the three following operations:

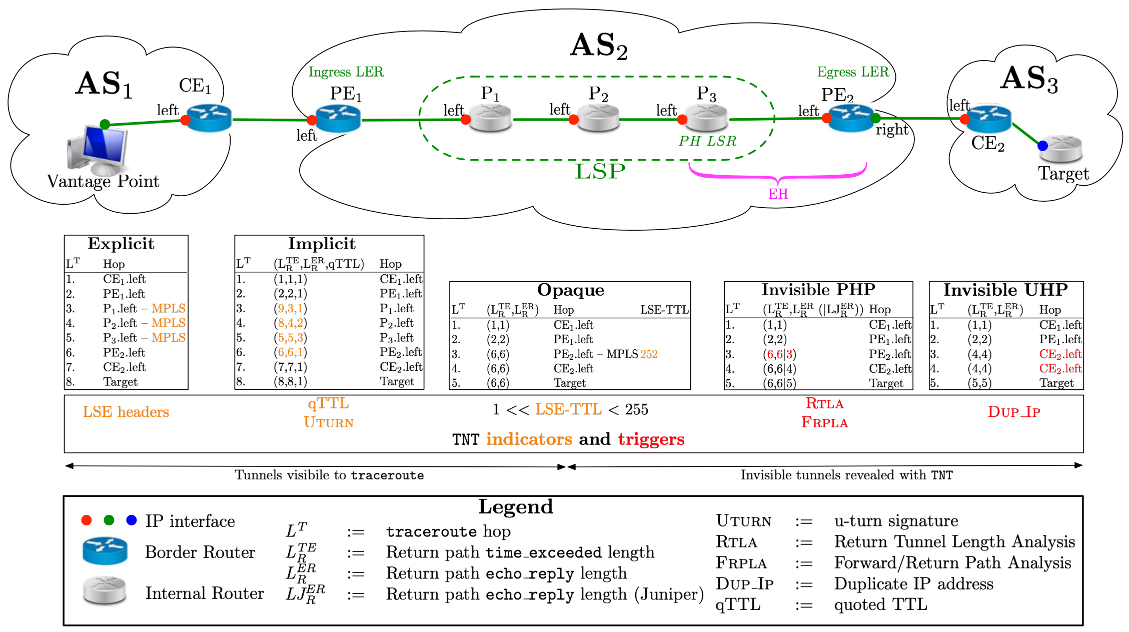
- PUSH (Sec. II-C1): the first MPLS router (the tunnel entry point) pushes one or several LSEs in the IP packet, turning it into an MPLS one. The *Ingress Label Edge Router* (Ingress LER) associates the packet FEC to its LSP;
- SWAP (Sec. II-C2): within the LSP, each LSR makes a label lookup in the LFIB, swaps the incoming label with its corresponding outgoing label, and sends the MPLS packet further along the LSP;
- POP (Sec. II-C3): the EH, the last LSR of the LSP, deletes the LSE, and converts the MPLS packet back into an IP one. The EH can be the *Egress Label Edge Router* (the Egress LER) when UHP is enabled or the PH LSR otherwise.

*1) LSP Entry Behavior (*PUSH*):* When an IP packet enters an MPLS cloud, the Ingress LER binds a label to the packet thanks to a lookup into its LFIB, depending on the packet FEC, e.g., its IP destination prefix. Before pushing the LSE into the packet, the Ingress LER has to initialize the LSE-TTL (see Fig. 1). Two behaviors can then be configured: either the Ingress LER sets the LSE-TTL to an arbitrary value (255, `no-ttl-propagate`) or it copies the current IP-TTL value into the LSE-TTL (`ttl-propagate`, the default behavior). Operators can configure this operation using the `no-ttl-propagate` option provided by the router manufacturer [22]. In the former case, the LSP is called a *pipe LSP*, while, in the latter case, a *uniform* one.

Once the LSE-TTL has been initialized, the LSE is pushed on the packet that is sent to an outgoing interface of the Ingress

[4]To simplify the presentation we will assume only one LSE in this section of the paper. This simplification is reasonable as the vast majority of collected tunnels only carry one label (i.e. more than 95% of the cases excluding VPRN usages).

[5]Note that, in the case of independent LSP control, any LSR can be in charge, despite itself, of the popping operation when the tunnel ends abruptly.



Explicit

L^T	Hop
1.	CE$_1$.left
2.	PE$_1$.left
3.	P$_1$.left – MPLS
4.	P$_2$.left – MPLS
5.	P$_3$.left – MPLS
6.	PE$_2$.left
7.	CE$_2$.left
8.	Target

Implicit

L^T	(L_R^{TE},L_R^{ER},qTTL)	Hop
1.	(1,1,1)	CE$_1$.left
2.	(2,2,1)	PE$_1$.left
3.	(9,3,1)	P$_1$.left
4.	(8,4,2)	P$_2$.left
5.	(5,5,3)	P$_3$.left
6.	(6,6,1)	PE$_2$.left
7.	(7,7,1)	CE$_2$.left
8.	(8,8,1)	Target

Opaque

L^T	(L_R^{TE},L_R^{ER})	Hop	LSE-TTL
1.	(1,1)	CE$_1$.left	
2.	(2,2)	PE$_1$.left	
3.	(6,6)	PE$_2$.left – MPLS 252	
4.	(6,6)	CE$_2$.left	
5.	(6,6)	Target	

Invisible PHP

L^T	(L_R^{TE},L_R^{ER} ($\vert LJ_R^{ER}$))	Hop
1.	(1,1)	CE$_1$.left
2.	(2,2)	PE$_1$.left
3.	(6,6$\vert$3)	PE$_2$.left
4.	(6,6$\vert$4)	CE$_2$.left
5.	(6,6$\vert$5)	Target

Invisible UHP

L^T	(L_R^{TE},L_R^{ER})	Hop
1.	(1,1)	CE$_1$.left
2.	(2,2)	PE$_1$.left
3.	(4,4)	CE$_2$.left
4.	(4,4)	CE$_2$.left
5.	(5,5)	Target

Legend

🔴 🟢 🔵	IP interface
	Border Router
	Internal Router

L^T	:=	traceroute hop
L_R^{TE}	:=	Return path time_exceeded length
L_R^{ER}	:=	Return path echo_reply length
LJ_R^{ER}	:=	Return path echo_reply length (Juniper)
UTURN	:=	u-turn signature
RTLA	:=	Return Tunnel Length Analysis
FRPLA	:=	Forward/Return Path Analysis
DUP_IP	:=	Duplicate IP address
qTTL	:=	quoted TTL

Fig. 2: Illustration of MPLS vocabulary and relationship between MPLS and `traceroute`. The figure is made of three parts. The upper part represents the network topology used throughout the paper to illustrate MPLS and TNT concepts. In particular, with respect to MPLS, P$_1$ is the LSP First Hop (FH), while P$_3$ is the Penultimate Hop LSR (PH LSR). In case of PHP, P$_3$ is the Ending Hop (EH) responsible for removing the LSE, while, with UHP, it is the Egress LER (PE$_2$). The middle part of the figure presents our MPLS classification. Finally, the bottom part of the figure provides triggers and indicators of an MPLS tunnel presence when probing with TNT. The relationship between the trigger/indicator and the observation made with probing is provided in red. Additional information (e.g. `time-exceeded` path length) are provided to illustrate TNT in Sec. V.

LER. In most cases, except for a given Juniper OS (i.e., Olive), the IP-TTL is decremented before being encapsulated into the MPLS header.

*2) LSP Internal Behavior (*SWAP*):* Upon an MPLS packet arrival, an LSR decrements its LSE-TTL. If it does not expire, the LSR looks up the label in its LFIB. It then swaps the top LSE with the one provided by the LFIB. The operation is actually a swap only if the outgoing label returned by the LFIB is neither Implicit NULL nor empty[6]. Otherwise, it is a POP operation as described in the next subsection. Finally, the packet is sent to the outgoing interface of the LSR with a new label, both according to the LFIB.

If the LSE-TTL expires, the LSR, in the fashion of any IP router, forges an ICMP `time-exceeded` that is sent back to the packet originator. It is worth to notice that an LSR may implement RFC 4950 [25] (as should be the case in all recent OSes). If so, the LSR will quote the full MPLS LSE stack of the expired packet in the ICMP `time-exceeded` message.

ICMP processing in MPLS tunnels varies according to the ICMP type of message. ICMP *Information messages* (e.g., `echo-reply`) are directly sent to the destination (e.g., the originator of the `echo-request`) if the IP FIB allows for it (otherwise no replies are generated). On the contrary, ICMP *Error messages* (e.g., `time-exceeded`) are generally forwarded to the Egress LER that will be in charge of forwarding the packet through its IP plane [8]. Differences between Juniper and Cisco OS and configurations are discussed in details in Sec. V-B

*3) LSP Exit Behavior (*POP*):* Upon the MPLS packet arrival, the EH decrements the LSE-TTL. If this TTL does not expire, the EH then pops the LSE stack after having determined the new IP-TTL.

Using PHP comes with the advantage of reducing the load on the Egress LER, especially if it is the root of a large reverse LSP-tree. Indeed, when using PHP, the last MPLS operation (i.e., POP) is performed one hop before the Egress LER, on the PH LSR. On the contrary, UHP[7] is generally used only when the ISP implements more sophisticated traffic engineering operations or wants to make the tunnel content and semantics more transparent to the customers (e.g., for VPRN purposes).

When a packet exits the tunnel, the router is left with a packet containing two TTLs: the IP-TTL, and the LSE-TTL. It thus has to decide which TTL should be kept and copied in the IP header before forwarding the packet as a standard IP packet. To ensure that the outgoing TTL cannot be greater than the incoming one, the EH would theoretically have to consider the configuration of the Ingress LER. If the Ingress LER has activated the `no-ttl-propagate` option, the EH should pick the IP-TTL of the incoming packet while the LSE-TTL should be selected otherwise. Indeed, in the former case, because the tunnel is hidden, the LSE-TTL was initialized at 255 and is likely superior to its IP counterpart. Consequently,

[6]In practice the actual label used for the forwarding is then greater than or equal to 0 (this specific value being reserved for Explicit NULL tunnel ending, i.e. for UHP) but excluding by design the reserved value 3 that is dedicated for Implicit NULL.

[7]The UHP feature has been recently made available on Juniper routers when LSPs are set with LDP. However, PHP remains the rule on Juniper [26, Chap. 1].

the EH should select the IP-TTL to ensure a monotonic decrement. In the latter case, the LSE-TTL was initialized at the value held by the IP-TTL, and is thus necessarily smaller than the IP-TTL upon exiting the tunnel as it now takes into account the MPLS hops. Consequently, the EH should here select the LST-TTL to ensure a monotonic decrement. In both cases, the TTL behavior remains monotonic. In order to synchronize both ends of the tunnel without any message exchange, two mechanisms might be used to select the IP-TTL at the EH:

1) applying a MIN(IP-TTL, LSE-TTL) operation (solution implemented for Cisco PHP configurations [24]), i.e., selecting the TTL which holds the smallest value;
2) assuming that the Ingress configuration (`ttl-propagate` or not) is the same as the local configuration (solution implemented by some JunOS and also in some Cisco UHP configuration).

Applying the MIN(IP-TTL, LSE-TTL) seems to be the best option, as it correctly supports heterogeneous `ttl-propagate` configurations while mitigating forwarding loops without exchanging signalization messages. This MIN(IP-TTL, LSE-TTL) operation might be used to detect the presence of hidden MPLS tunnels [9]. Indeed, it is likely that the ICMP `time-exceeded` message generated by the EH will enter the same MPLS cloud immediately to reach the vantage point.

In that case, when the reply leaves the MPLS cloud, its IP-TTL will not have been decremented, while the LSE-TTL will take the number of hops within the MPLS tunnel into account. Consequently, the EH of the return path (P_1 in Fig. 2) will choose to copy the LSE-TTL in the IP-TTL, as the IP-TTL of the reply still holds its maximum value (255 for a Cisco router – see Sec. II-A). Thus, while the forward path through the hidden MPLS cloud has no effect on the IP-TTL of the packet, the return path is taken into account, as the PH LSR of the return path (P_1), copies the LSE-TTL within the IP-TTL.

It is interesting to mention that this MPLS behavior strongly depends on the implementation and configuration. For instance, on some Juniper OS routers or when the UHP option is activated on some Cisco IOS, the MIN(IP-TTL, LSE-TTL) operation is not systematically applied. The EH assumes an homogeneous propagation configuration among LERs. When it is not the case (`ttl-propagate` at one end of the tunnel and `no-ttl-propagate` at the other end), the EH will use the IP-TTL instead of the LSE-TTL, leading to a so-called *jump* effect with `traceroute`. In other words, as many hops as the LSP length are skipped after the tunnel by `traceroute`, the TTL of the packet is brought back to the value it held before going through the LSP. Except when explicitly stated, we will consider homogeneous configurations (e.g., `ttl-propagate` on the whole tunnel) in the remainder of the paper. Finally, it is worth noticing that mixing UHP and PHP (hybrid configurations) can also result in uncommon behaviors.[8]

[8]Those behaviors are described and discussed in details in a companion technical report [27].

III. REVISITING MPLS TUNNELS TAXONOMY

According to whether LSRs implement RFC4950 (i.e., ICMP `time-exceeded` quoting MPLS LSE) or not and whether they activate the `ttl-propagate` option or not, MPLS tunnels are more or less visible to `traceroute` [8].

Explicit tunnels are tunnels with RFC4950 and the `ttl-propagate` option enabled. As such, they are fully visible with `traceroute`, including the labels used along the LSP. ***Implicit*** tunnels also enable the `ttl-propagate` option but do not implement the RFC4950. IP level information is not missing but LSRs are seen as ordinary routers; leading to a lack of "semantic" in the `traceroute` output. ***Opaque*** tunnels are partially obscured from `traceroute` as the `ttl-propagate` option is disabled while the RFC4950 is implemented. Moreover, an Opaque LSP ends at its EH with a non-terminating label. Consequently, the EH is the only hop being seen as an MPLS one while the internal content of the LSP is totally hidden. Finally, ***Invisible*** tunnels are fully hidden as the `no-ttl-propagate` option is enabled and the LSP ends properly (RFC4950 being implemented or not).

As illustrated in Fig. 2, **Explicit tunnels** constitute the ideal case as all the MPLS information comes natively with `traceroute`. For **Implicit tunnels**, Donnet et al. [8] have proposed techniques to identify their LSRs based on the way they process ICMP messages and the quotation of the IP-TTL in the `time-exceeded` reply (qTTL and UTURN in Fig. 2).

Opaque tunnels are only encountered with Cisco LSPs and are due to LSPs ending abruptly, in an improper fashion. In other words, the MPLS packet reaches the exit point of the tunnel without a terminating label (Implicit or Explicit NULL) within its LSE to properly signal the end of the LSP, causing the LSP to *break*. Thanks to our large scale campaign and experiments with our emulation platform, we conclude that the vast majority of Opaque tunnels are caused by Carrier-of-Carriers VPN [28] or similar technologies. Indeed, such technologies provoke an abrupt tunnel ending as the LSP ends with the LSE containing the label used to identify the VPN instead of a standard terminating label. As we will show later in details, they lead to non-revealable tunnels.

The `traceroute` behavior for Invisible tunnels differs according to the popping scheme (i.e., PHP or UHP) and the OS, as illustrated in Fig. 2. While **Invisible PHP** tunnels are identified through path length asymmetry [9] (see Sec. V), Invisible UHP tunnels provoke a duplicated IP (at least with the IOS 15.2). More precisely, upon the reception of a packet having an IP-TTL of 1, the Egress LER (PE_2 in Fig. 2) does not decrement this TTL, but rather forwards the packet to the next hop (CE_2 in the example), leading so to the Egress being hidden in the trace. In contrast, the next hop will appear twice: once for the probe that should have expired at the Egress and once at the next probe. This surprising pattern, a duplicated IP at two successive hops, illustrated as **Invisible UHP** in Fig. 2 might be misunderstood as a forwarding loop.

IV. HIDDEN TUNNEL REVELATION

Techniques for revealing the content of Invisible PHP and UHP tunnels are similar. In the case of an Invisible PHP tunnel, they can be applied directly as we know both ends of the tunnel (Ingress and Egress LER – see Fig. 2). However, for Invisible UHP, the Egress LER is missing from the `traceroute` output (look at middle part of Fig. 2).

It is nevertheless possible with Invisible UHP to infer the outgoing IP interface of the Egress LER (the right interface, in green, on PE_2 in Fig. 2). Thanks to its retrieval, `TNT` can force replies from the Egress LER incoming interface (the left one, in red, on PE_2 in Fig. 2). This technique, called *buddy*, assumes a simple point-to-point connection between the Egress LER and its next-hop (this naive assumption comes for the sake of simplicity, but the technique can be extended to deal with point-to-multipoint subnet [4, 29, 30]). The IP addresses belonging to the same /31 or /30 prefix are called **buddies** and `TNT` just needs to infer the correct prefix length to guess the address of CE_2's buddy (i.e., PE_2.right in Fig. 2).

With a /30, four IP addresses are available: addresses 0 and 3 are the network and broadcast addresses while addresses 1 and 2 are used for numbering interfaces. If CE_2.left corresponds to address 0 (resp. address 3) in a /30, it means that PE_2 and CE_2 share a /31 and PE_2.right is address 1 (resp. address 2) of the /30. However, if CE_2.left corresponds to address 1 (resp. address 2), we launch a `ping` towards address 0 within the /30. If an `echo-reply` is received, both interfaces are on a /31 and PE_2.right corresponds to address 0 (resp. address 3). Otherwise, both interfaces are on a /30 and PE_2.right corresponds to address 2 (resp. address 3 if CE_2.left corresponds to address 2). Note that the buddy identification process can be further improved by considering more advanced techniques [31] whose probing overhead can be mitigated.

As ICMP `time-exceeded` typically contains the IP address of the incoming interface having received the expiring probe, running a `traceroute` towards the inferred address of PE_2.right allows to obtain PE_2.left. Once the potential Ingress and Egress LERs are known, we can launch a hidden tunnel revelation technique, i.e., DPR or BRPR [9]. The choice of technique depends on the way labels have been bound to destination prefixes (see Sec. II-B). It is worth recalling that one can easily discriminate Cisco and Juniper devices using network fingerprinting [19].

On the one hand, with ordered LSP control used with Juniper by default on loopback addresses, all the external BGP transit traffic goes through MPLS tunnels while the traffic destined to internal prefixes relies on IP forwarding. Thus, a single `traceroute` targeting the internal Egress LER is enough to reveal all LSRs along the LSP. This technique is called ***Direct Path Revelation*** (DPR). Applying DPR on Fig. 2, `TNT` simply sends probes targeting PE_2 revealing P_1, P_2, and P_3 in a single shot (without labels, as the probe targeting PE_2 follows the same path as a transiting probe, but without entering the MPLS cloud).

On the other hand, with independent LSP control used by Cisco by default on all IP addresses, LDP is entirely enabled for all the network such that each LSR binds labels for each prefix in its IGP RIB. Thus, as all traffic goes through the MPLS cloud, DPR can not be used. Our other revelation technique, ***Backward Recursive Path Revelation*** *(BRPR) takes benefit from the prefix locality: the targeted incoming interface of the Egress LER is in the same prefix as the outgoing interface of the PH LSR.* Thus, since the PH LSR is directly connected to the targeted prefix, it acts as the Egress LER for it and consequently becomes visible to `traceroute`. Applying this method iteratively in a backward fashion up until the Ingress LER, we can reveal each hidden LSR. Applying BRPR on Fig. 2, we first send a `traceroute` towards PE_2 and discover P_3. We next send a `traceroute` towards P_3 and discover P_2 and so on until the Ingress LER is met again.

As the targeted IP changes at each iteration of BRPR, its outcome may be affected by load balancing. Revealed links may not belong to the same consistent path. Conversely, DPR works in a single shot and does not suffer from this limit (as `TNT` is built upon Paris Traceroute which relies on constant five tuples in each probe of the same trace).

V. TNT Design

This section introduces our tool, `TNT` (**T**race the **N**aughty **T**unnels), able to reveal most of MPLS tunnels hidden along a path. `TNT` is an open-source scamper [11] plugin extension built upon Paris Traceroute [12], in order to mitigate load balancing issues.

`TNT` consists in collecting, in a hop-limited fashion, intermediate IP addresses between the vantage point and a given target. The tracing phase ends when the target has been reached or a gap has been encountered (e.g., five consecutive non-responding hops). `TNT` uses a moving window of two hops such that, at each iteration, it looks for <Ingress/Egress> pairs of candidates, possibly hiding Invisible tunnels.

For each pair of collected IP addresses, `TNT` checks for the presence of tunnels through so-called *indicators* and *triggers*. The former provides reliable indications about the presence of an MPLS tunnel without requiring additional probing. **Indicators** suggest uniform tunnels, and are basic evidence of visible MPLS presence such as LSEs quoted in the ICMP `time-exceeded` packet (see Sec. V-A1 for details and exceptions). **Triggers**, except DUP_IP, consider unsigned values suggesting the presence of Invisible tunnels through a large shifting in path length (see Sec. V-A2 for more details). When exceeding a given threshold $\mathcal{T}$, a revelation is attempted as already developed in Sec. IV. `TNT` is cautious by design: we do not conclude anything from revelations or detections hindered by network anomalies. In addition, while `TNT` is, as other active probing tools, subject to network anomalies, we designed it to be fairly resilient to load balancing and rate limiting thanks to its Paris Traceroute base and inherent lightweight nature respectively.

Fig. 2 highlights the main patterns `TNT` looks for in a simple scenario where forward and return paths are symmetrical.

A. Indicators and Triggers

Listing 1: Pseudo-code for checking indicators and triggers

```
if (is_mpls(cur_hop))
  if ($\mathcal{T}_{LSE\_TTL}$ < cur_hop.lse_ttl < 255)
    return LSE−TTL #Opaque tunnel
  else
    return LSE #Explicit tunnel
```

```
 7 if (cur_hop.qttl > 1)
 8     return qTTL #Implicit tunnel
 9
10 if (cur_hop == next_hop)
11     return DUP_IP #Invisible UHP tunnel
12
13 #inferring path length from raw TTLs
14 $L_R^{TE}$ = path_len(cur_hop.ttl_te)
15 $L_R^{ER}$ = path_len(cur_hop.ttl_er)
16 $L^T$ = cur_hop.probe_ttl
17 diff_te_er = $L_R^{TE} - L_R^{ER}$
18
19 if (sign_is_junOS(cur_hop))
20     if (diff_te_er $\geq \mathcal{T}_{\text{RTLA}}$)
21         return RTLA #Invisible PHP tunnel
22 elif (|diff_te_er| $> \mathcal{T}_{\text{UTURN}}$)
23         return UTURN #Implicit tunnel
24 if ($L_R^{TE} - L^T \geq \mathcal{T}_{\text{FRPLA}}$)
25     return FRPLA #Invisible PHP tunnel
```

Listing 1 provides the pseudo-code for checking indicators and triggers such as implemented in TNT.

1) Visible Tunnel Indicators: They are pieces of evidence of an MPLS tunnel presence and concern cases where tunnels (or parts of them) can be directly retrieved from the traceroute output. Explicit tunnels are indicated through LSEs directly quoted in the ICMP time-exceeded message – See line 5 in Listing 1 and traceroute output on Fig. 2.

The indicator for Opaque tunnels consists of a single hop LSP with a quoted LSE-TTL not being equal to an expired value. This abnormal behavior is due to the way labels are handled with Cisco routers, in particular with VPRN tunnel ending. This is illustrated in Fig. 2 where we get a value of 252 because the LSP is actually 3 hops long. This surprising quoted LSE-TTL is evidence in itself. It is illustrated in lines 2 to 3 in Listing 1. A single hop is tagged as Opaque if the quoted LSE-TTL is between a minimum threshold (Sec. VI discusses its calibration), $\mathcal{T}_{LSE\_TTL}$ and 254 (the LSE-TTL being initialized to 255). This is the only indicator that can fire additional probing in order to reveal the content of the tunnel. However, we will explain in the next section why, in practice, it does not perform well as a trigger.

Implicit tunnels are detected through qTTL and/or UTURN indicators [8]. First, if the IP-TTL quoted in an ICMP time-exceeded message (qTTL) is greater than one, it likely reveals the ttl-propagate option at the Ingress LER of an LSP. As the LSE-TTL was initialized at the IP-TTL value, the packet can expire within the LSP. However, as the IP-TTL is not decremented within the tunnel, the qTTL is greater than one. For each subsequent traceroute probe within the LSP, the qTTL will be one greater, resulting in an increasing sequence of qTTL values. This indicator is considered in line 7 in Listing 1. Second and by default, the UTURN indicator relies on the fact that LSRs send ICMP time-exceeded messages to the Egress LER which, in its turn, forwards the packets to the probing source. However, such LSR reply directly to other kinds of probes (e.g., echo-request) using their own IP forwarding table, if available. As a result, return paths are generally shorter considering echo-reply messages than regarding time-exceeded replies. Thereby, the UTURN indicator reflects this difference in these lengths. Note that while the UTURN and RTLA computations are identical, Juniper routers do not exhibit, by default, any implicit UTURN pattern. Consequently, even though some configurations could enable this pattern on Juniper routers, we do not consider this case.

2) Triggers for Revealing Invisible Tunnels: They are patterns suggesting their presence (both for Invisible PHP and UHP) and so firing additional probing (see Sec. IV). TNT looks first for potential Invisible UHP tunnels (line 10). As explained in Sec. III, they occur with Cisco routers using IOS 15.2 and result in a duplicate IP address in the trace output (CE_2 in Fig. 2).

The two remaining triggers, RTLA (Return Tunnel Length Analysis) and FRPLA (Forward/Return Path Analysis) [9], rely on path lengths. More precisely, RTLA is the difference between the time-exceeded and the echo-reply return path lengths, while FRPLA is the difference between the forward and the return path lengths of traceroute probes and associated replies. Both triggers are based on the idea that replies sent back to the vantage point are also likely to cross back the MPLS cloud, which will lead to the application of the MIN(IP-TTL, LSE-TTL) operation at the EH of the return tunnel. In the absence of Invisible tunnels, we expect to find length differences equal or close to 0. Therefore, any significant deviation[9] from this value is interpreted as the potential presence of an Invisible MPLS cloud, and thus, fires additional path revelation techniques (see Sec. IV).

To check for those triggers, we first extract the key distances thanks to the IP-TTLs in replies received by the vantage point (lines 14 to 16 in Listing 1). Since RTLA only works with JunOS routers [9], prior to estimating the triggers, TNT uses network fingerprinting [19] to determine the router brand of the potential Egress LER (line 19 in Listing 1).

In the presence of a JunOS hardware (line 19), time-exceeded and echo-reply packets have different initial TTL values [19], and the RTLA trigger can exploit the TTL gap between those two kinds of messages caused by the MIN(IP-TTL, LSE-TTL) behavior at the Egress LER. Indeed, the L_R^{ER} is longer than the LJ_R^{TE} as the MIN operation considers a differentiated pick. This difference represents the number of LSRs in the return LSP, and is compared to a predefined threshold $\mathcal{T}_{\text{RTLA}}$(line 20). This threshold (see Sec. VI for the parameter calibration) filters out very short LSPs. Finally, if the signature does not correspond to JunOS, TNT falls back to the UTURN indicator (see line 23).

FRPLA is more generic and applies thus to any configuration. FRPLA compares the lengths of the forward (i.e., L^T) and return paths (i.e., L_R^{TE}). In the presence of MPLS tunnels, return paths are expected to be seen as longer than forward ones. Indeed, LSRs are not counted in the forward path while they are taken into account in the return paths due to the MIN(IP-TTL, LSE-TTL) behavior at the return Egress LER. Then, we can analyze their length difference and check if a shift appears (see Line 24). This is illustrated in Fig. 2 ("Invisible PHP") in which L^T is 3 while L_R^{TE} is equal to 6, leading so to an estimation of the return tunnel length of 3. At the AS granularity, when no IP hop is hidden, we expect that the values associated to FRPLA will look like a normal

[9]In practice, we do not consider negative values. Indeed, they do not suggest the presence of MPLS tunnels but rather path asymmetry evidences (for FRPLA) or load balancing practices on the return path (for RTLA).

Configurations	Pop	Cisco iOS15.2	Juniper VMX
P2P circuits (e.g., LDP or RSVP-TE tunnels)	PHP	FRPLA, BRPR ☑☒	RTLA, DPR ☑
	UHP	DUP_IP, BRPR++ ☑☒	RTLA, DPR ☑
P2MP overlays (e.g., VPRN: CsC or VPN BGP-MPLS)	PHP	LSE-TTL, - ☒	RTLA++, - ☒
	UHP	LSE-TTL++, - ☒	N/A

TABLE III: `TNT` revelation (☑) and classification (☒) capacities according to the OS and the MPLS tunneling technologies (P2P or P2MP). This table also provides the default indicator/trigger and its associated path revelation method.

distribution centered in 0 (i.e., forward and return paths have, on average, a similar length). If we rather observe a significant and generalized shift towards positive values, it means the AS probably makes use of the `no-ttl-propagate` option. As it relies on the difference in length of return paths, RTLA is resilient to path asymmetry. FRPLA, however, relying on the difference between a forward and return path, is more sensible to it. To handle path asymmetry at the trace granularity, `TNT` uses a threshold, $\mathcal{T}_{\text{FRPLA}} > 0$, to avoid generating numerous false positives.

The main purpose of triggers is to limit the overhead generated by `TNT`. Revelations launched at each hop in a brute force fashion may reveal nearly all MPLS tunnels. However, by first checking for triggers, we limit the amount of unnecessary probes (i.e., leading to no revelation).

B. `TNT` *Limits and Opaque Tunnels*

By using GNS3, we aimed first at verifying that the inference assumptions considered in the wild are correct and reproducible under a controlled environment, validating so the triggers, indicators, and revelation methods used by `TNT`. Second, some of the phenomena we exploit to reveal tunnels in the wild have been directly discovered in our testbed by reverse-engineering the TTL processing of some common OSes used by many real routers. Indeed, our emulated testbed allowed us to run several OSes and numerous configurations in a controlled environment, similarly to a physical testbed. Thus, we could link each triggers and indicators to specific kinds of tunnels as well as establish the limits of `TNT`. All details and results of experiments done with GNS3 are provided in the companion technical report of this paper [27].

Table III provides a summary of `TNT` revelation and discrimination capacities considering several MPLS usages in standard configurations. In particular, it shows that `TNT` is able to discriminate between Cisco Invisible UHP and PHP tunnels while it is not the case for Juniper routers. Indeed, for both UHP/PHP Juniper configurations, the trigger and the revelation methods are the same (RTLA and DPR respectively). Moreover, we also show for which cases our basic set of techniques needs to be extended for enabling revelation and distinction among different classes. We use the symbol $++$ to highlight these new requirements. For example, revealing UHP Cisco tunnels requires to extend BRPR with the additional buddy functionality (see Sec. IV) and UDP[10] probing in order to extract the incoming Egress IP address that, in turn, allows `TNT` to reveal the tunnel. LSE-TTL++ refers to a way of discriminating UHP VPRN from PHP ones, both resulting in Opaque tunnels (with UHP, the quoted LSE-TTL is equal to 255 instead of reflecting the length of the tunnel). Finally, RTLA++ is a way to distinguish VPRN configuration from basic tunneling on Juniper devices. We discuss this specific situation at the end of the section as it is more complex.

Generally speaking, Opaque tunnels may arise for different reasons, such as routing devices heterogeneity, BGP edge configurations, or VPRN. Our GNS3 platform shows that VPRN content cannot be revealed with `TNT`, while other Opaque tunnels can. However, both arise from a non-standard terminating label. Indeed, upon its arrival at the Egress, at least one label is still present in the MPLS header. This surviving inner label is used to identify the VPN and the associated VRF[11]. As the VPN label value is neither Explicit NULL nor Implicit NULL, the Egress behaves as if the tunnel did not end in a controlled fashion.

This absence of content revelation can be explained by the IP address collected by `TNT` from the ICMP reply. Usually, this address is the one of the incoming interface of the Egress PE. In the Cisco VPRN case, the collected IP address is the one assigned to the interface onto which the VRF is attached which usually is the outgoing interface, towards the VPN at the customer's side (see Sec. VII for details). Because the incoming address is the only one that enables a successful revelation, this type of Opaque tunnels cannot be revealed. While the outgoing address usually allows `TNT` to get the incoming one, it turns out to be impossible within a VPRN, as all probes are pushed to the VRF of the VPN and its associated interface before the error message is generated.

Juniper VPRNs behave in a slightly different fashion. For such tunnels, no Opaque indicator can be seen. Instead, similarly to Cisco Invisible UHP tunnels, the packets destined to the VPN are IP forwarded directly to the next-hop without manipulating or looking at the IP-TTL whatever its value. Thus, when performing a direct trace targeting the IP interface of the Egress LER belonging to the VPN, this address and its buddy appear in the wrong order. The two addresses are switched, meaning that the CE IP address appears before the Egress one. Indeed, being forwarded without inspecting the IP-TTL, the probes targeting an IP belonging to the VPN are automatically forwarded to the corresponding CE router where they expire. The next probe, having a greater initial TTL, follows the same path, but can be forwarded back to the Egress, its destination, by the CE router. This about-turn can be inferred as the two IP addresses are switched regarding their actual position, and the TTL of the ICMP `time-exceeded` deviates from its strict monotony (as the first probe went further than the second one). These two artifacts are reflected by the RTLA++ method in Table III.

While RTLA++ can theoretically discriminate Juniper VPRN from basic P2P circuits, this extended trigger would be fairly unreliable in practice, as the artifacts it tries to detect

[10]With ICMP probes, the target will not answer with its incoming IP address as the probe does not result in a error reply when reaching the target.

[11]In case of VPRN, a router contains a Virtual Routing and Forwarding table (VRF) for each virtual network.

are minute compared to the Opaque indicator. However, RTLA being itself a pretty reliable trigger for Juniper devices, it should consequently always result in the revelation of internal LSRs. Thus, following an RTLA trigger, if no new content is revealed while the Ingress was reached, one can conclude at a Juniper VPRN.

VI. TNT Calibration

As shown in the previous section, `TNT` relies mainly on four thresholds related to indicators and triggers to limit its overhead: $\mathcal{T}_{LSE\_TTL}$ for Opaque tunnels, $\mathcal{T}_{\text{UTURN}}$ for Implicit tunnels, and both $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$ for Invisible ones. This section aims at experimentally calibrating those thresholds, as well as evaluating the probing cost of our approach in general. We do not aim here at validating `TNT`, but merely calibrating it in order to make it more effective in terms of probing cost.

A. Calibration Setup

For this specific calibration analysis, we deployed `TNT` over a limited number of vantage points (VPs) as such experiments are costly due to the brute force approach detailed below. In practice, we consider only 3 VPs over the Archipelago infrastructure [17]. They were located in Europe (Belgium), North America (San Diego), and Asia (Tokyo). `TNT` was run on April 6th, 2018 towards a set of 10,000 destinations (randomly chosen among the whole set of Archipelago destinations list). Each VP had its own list of destinations, without any overlapping.

First, we have observed that abnormal[12] LSE-TTL values vary between 236 and 254. Consequently, a value of 236 for $\mathcal{T}_{LSE\_TTL}$ would be enough for detecting the presence of an Opaque tunnel. Lower values are considered as anomalies.

Besides, from indicators and triggers described in Sec. V-A, one can observe that the UTURN computation is equivalent to RTLA for Juniper routers (in practice, only RTLA is considered for Juniper routers as they do not exhibit such an Implicit pattern by default on our testbeds). The $\mathcal{T}_{\text{UTURN}}$ value used for Cisco implicit tunnels does not need to be the same value than $\mathcal{T}_{\text{RTLA}}$ for Juniper routers. By design, $\mathcal{T}_{\text{UTURN}} = 0$ as any difference between `echo-reply` and `time-exceeded` replies for the Cisco router signature indicates a LSE-/IP-TTL shifting. In practice, we reinforce the condition by looking for at least two consecutive hops having a cumulated UTURN ≥ 3.

Finally, for thresholds $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$, we scanned all values between 0 and 4. A full calibration campaign was launched for each pair of thresholds. For each pair, if no trigger is pulled, a so-called brute force revelation is undertaken: DPR/BRPR are then launched (with the use of the buddy when required) in any case. This brute force data is used as a basis to evaluate the quality and cost of each threshold value.

Indeed, due to the reliability of our revelation methods (taking advantage of the inner workings of MPLS), we calibrate our triggers by checking if they accurately reflect the data `TNT` produces when used in a brute force fashion (i.e., revelation is fired at each hop and if nothing is revealed, we consider that there is no tunnel).

[12]Abnormal here means "different from 1 or 255" which is the LSE-TTL value that should be obtained in ICMP `time-exceeded` messages. More details can be found in our technical report [27].

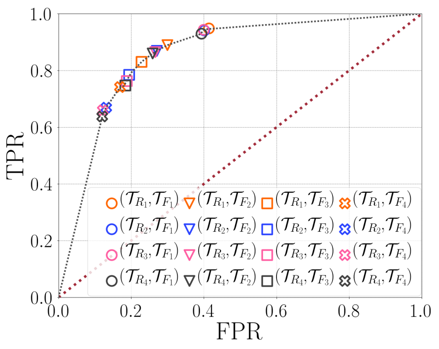


Fig. 3: Receiver operating characteristic (ROC) curve providing the performances of `TNT` according to the thresholds used for revealing Invisible tunnels. $\mathcal{T}_{R_x}$ refers to $\mathcal{T}_{\text{RTLA}}$with the value x, while $\mathcal{T}_{F_y}$ to $\mathcal{T}_{\text{FRPLA}}$with the value y.

B. Calibration Analysis

The two explored triggers, namely FRPLA and RTLA, and their associated thresholds provide a certain prediction, while the results of additional probing give reliable facts, i.e., if a tunnel is present or not by showing new IP hops. This binary classification allows us to assess the performances of FRPLA and RTLA according to the calibration of their thresholds. We evaluate their conjoint performances through the analysis of True Positive Rate (TPR) and False Positive Rate (FPR). We plot the results on a Receiver Operating Characteristic (ROC) curve in Fig. 3. We define TPR as the ratio of `TNT` success to the number of links actually being MPLS tunnels (having a length greater than 1). In such a case, `TNT` correctly triggered additional probing for revealing Invisible tunnels. We then have $TPR + FNR = 1$, i.e., when adding to False Negative Rate (tunnels revealed only with the brute force approach), we obtain all links being long enough tunnels. FPR is defined as the ratio of `TNT` failure to the number of standard IP links: additional probing was triggered but without revealing anything. We have $FPR + TNR = 1$, i.e., when adding to True Negative Rate, that is IP links where nothing can be discovered despite additional probing, we obtain all IP links without tunnels. In practice, False Negatives and False Positives are only an issue when considering the use of triggers only, without relying on additional revelation launches. Indeed, False Negatives may be filtered out through the use of a brute force approach or using lenient trigger thresholds. Similarly, False Positives may be filtered out as the triggered revelation will not reveal any hidden hops.

Our ROC curve has been plotted considering the $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$ thresholds between 1 and 4. The red dotted diagonal provides the separation between positive results for `TNT` (above part of the graph) and negative results (below part of the graph). Finally, the black dotted line is the interpolation of experimental results.

We observe that the results are essentially positive for `TNT`. Considering the couples ($\mathcal{T}_{R_1}$, $\mathcal{T}_{F_3}$) and ($\mathcal{T}_{R_2}$, $\mathcal{T}_{F_3}$), performances are close to a pretty good classification (upper

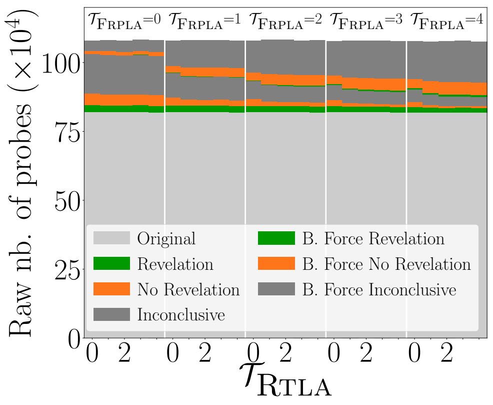


Fig. 4: Probing cost associated to TNT according to $\mathcal{T}_{\text{FRPLA}}$and $\mathcal{T}_{\text{RTLA}}$thresholds. Here we also evaluate their uses with a value of 0 leading to 25 combination pairs (instead of 16 previously). The X-Axis plots, for multiple $\mathcal{T}_{\text{FRPLA}}$(see labels above) the corresponding $\mathcal{T}_{\text{RTLA}}$($\in [0, 4]$).

left corner). These couples are thus considered as the best choice for defining our thresholds $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$. We obtain a compromise close to 80%-20%: while we expect to reveal $\approx$ 80% of existing tunnels (MPLS links), TNT has a controlled overhead of 20%, i.e., it only fires useless additional probing for an average limited to two actual IP links out of ten. One can reach a higher TPR (close to 95% for $\mathcal{T}_{F_1}$), leading to a higher FPR (close to 40%). However, as mentioned before, these False Positives do not distort the resulting statistics, but merely lead to a higher (but still limited) overhead as more revelations will be launched.

Since our triggers detected most MPLS tunnels that were revealed through brute force probing, we can conclude that our set of detection methods covers by themselves the detection of a large share of MPLS tunnels (and so limit the extra probing cost of revelations).

C. Probing Cost

Fig. 4 illustrates the probing cost associated to TNT. In particular, it focuses on additional measurements triggered by RTLA or FRPLA for revealing Invisible tunnels. The light grey zone (labeled as "Original" on Fig. 4) corresponds to probes associated to standard `traceroute`. The green, orange, and dark grey zones correspond to probes sent when additional measurements are triggered by RTLA or FRPLA. In particular, the green zone corresponds to additional measurements that were able to reveal the content of an Invisible tunnel. On the contrary, the orange zone refers to additional measurements that failed, i.e., no Invisible tunnel content was revealed. Finally, the dark grey zone refers to inconclusive revelation: the trigger has led to additional measurements but TNT was unable to reach the potential Egress LER (i.e., the IP address that engaged the trigger generally due to unresponsive IP interfaces) or TNT was unable to reach again the candidate Ingress LER because the path has changed (ECMP or BGP routing noises).

The amount of probes linked to successful revelations (green) remains almost stable whatever the values for $\mathcal{T}_{\text{FRPLA}}$ and $\mathcal{T}_{\text{RTLA}}$ are. However, a very slow and limited decrease occurs for high values (some tunnels are missed when using too conservatives thresholds). Further, the additional traffic generated by erroneous triggers (orange) or by inconclusive revelation (dark grey) decreases while $\mathcal{T}_{\text{FRPLA}}$increases. This result is aligned with Sec. VI-B in which the best values for $\mathcal{T}_{\text{FRPLA}}$ are between 1 and 3. Note that FRPLA is more generic but less reliable than other triggers, thus using more conservatives thresholds quickly limits some of the noise. On the contrary, the $\mathcal{T}_{\text{RTLA}}$ threshold has a minor effect on the amount of probes sent as the associated trigger is more reliable and specific.

Hatched zones (orange, dark grey, and green) correspond to the amount of probes sent using brute force. The results showcased on Fig. 3 are here shown in a more practical fashion, and exhibit two interesting ways of calibrating TNT. The more conservative configuration ($\mathcal{T}_{\text{RTLA}}$=2 and $\mathcal{T}_{\text{FRPLA}}$=3) leads to a slightly higher number of undetected tunnels (dashed green) compared to its more lenient counterpart ($\mathcal{T}_{\text{RTLA}}$=1 and $\mathcal{T}_{\text{FRPLA}}$=1), but reduces the number of unsuccessful launched revelations (dashed orange and grey). The more lenient thresholds increase the number of probes used (plain), but allows to detect about 95% of all tunnels seen (green).

Generally speaking, considering the information gathered, one can observe that the overhead of TNT is limited compared to a standard active campaign. In particular, using correct thresholds to limit both useless probes and missed tunnels (e.g., $\mathcal{T}_{R_1}$, $\mathcal{T}_{F_3}$), our tool generates less than 10% of additional probing compared to the underlying campaign and reaches a satisfying compromise where $\approx$ 80% of tunnels are revealed.

VII. Tunnels Quantification With TNT

This section aims at discussing the capacities of TNT in the wild Internet. In particular, it analyzes the relative coverage of each indicator and trigger with respect to possible revelation techniques. Sec. VII-A describes the measurement setup, while Sec. VII-B discusses the results obtained.

A. Measurement Setup

We deployed TNT on the Archipelago infrastructure [17] on April 23rd, 2018 with parameters $\mathcal{T}_{\text{FRPLA}}$fixed to 3 and $\mathcal{T}_{\text{RTLA}}$to 1, according to results discussed in Sec. VI-B.

TNT has been deployed over 28 vantage points, 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 IP addresses, is inherited from the Archipelago dataset and spread over the 28 vantage points to speed up the probing process.

A total of 522,049 distinct unique IP addresses (excluding `traceroute` targets) have been collected, with 28,350 being non-publicly routable addresses (and thus excluded from our dataset). Each collected routable IP address has been pinged once per vantage point, allowing us to collect additional data for fingerprinting (see Sec. II-A). Our dataset and our post-processing scripts are freely available.[3]

Fig. 5 shows the proportion of paths, per monitor, that crosses at least one LSP. We see that, for more than half of the monitors, at least 50% of the paths include one MPLS

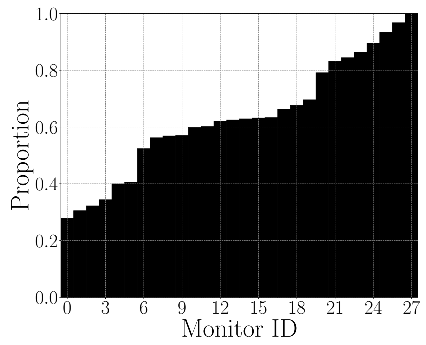


Fig. 5: Paths, per monitor, crossing at least one LSP.

Status		# probes		
		traceroute	ping	buddy
original		63,559,385	7,109,075	–
attempt	revealed	2,190,275	206,842	19,181
	no revelation	1,640,224	–	556
	TARGET_NOT_REACHED	4,174,404	–	9,888
	ING_NOT_FOUND	1,790,900	–	7,326

TABLE IV: Raw number of probes sent by TNT over the set of 28 vantage points.

tunnel or more. This echoes previous work on MPLS large deployment [8, 10, 32].

B. Results

Table IV provides the amount of probes sent by traceroute-like probing in TNT, ping, and buddy bit exploration. The row "original" refers to standard traceroute (i.e., actual IP links, Explicit, or Implicit tunnels).

The main outcome of Table IV is the amount of probes involved in inconclusive revelations, i.e. TARGET_NOT_REACHED cases (TNT was unable to reach the potential Egress LER) and ING_NOT_FOUND cases (TNT did not cross the potential Ingress LER). In particular, TARGET_NOT_REACHED generate almost twice more probes than revealed tunnels. Those particular inconclusive revelations might be explained by ICMP rate limiting due to additional probing. Another explanation is that those potential Egress LERs respond to initial traceroute with an IP address that is not globally announced. As such, no route is available to reach them. The potential Egress LER turned out to be unreachable for about 40% of all the attempted revelations. While the revelations were inconclusive, this result can be seen as an evidence of MPLS usage, as it may be due to a router possessing an incomplete IP routing table, or operator policies aiming to hide their network.

Table V provides the number of MPLS tunnels discovered by TNT, per tunnel class as indicated in the first column. The indicators/triggers are provided, as well as the additional revelation technique used. Explicit tunnels are the most prevalent class (76% of tunnels discovered): most operators do not seem to hide their MPLS infrastructure.

Implicit tunnels represent 5% of the whole dataset, with the UTURN indicator being more present than the qTTL one. Compared to previous works, it is clear that this class is not as prevalent as expected at the time, both because we corrected and improved our methodology by defining RTLA for Juniper routers, and also because the RFC4950 is likely to be more and more deployed.

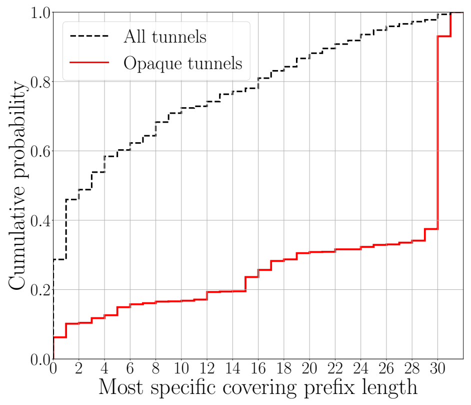


Fig. 6: Distribution of most specific prefix covering pair (Egress LER,next-IP).

Opaque tunnels are less prevalent (1.7% of tunnels discovered). Additional revelation techniques (DPR or BRPR) do not perform well with such tunnels. The content of 98% of Opaque tunnels cannot be revealed, suggesting that the vast majority of Opaque tunnels arise due to Cisco VPRNs. This is confirmed by Fig. 6 that plots, as a CDF (Y-Axis), the distribution of most specific prefix length (X-Axis) covering the pairs (Egress LER, next-IP). We call "next-IP" the IP address following the Egress LER in the trace. The black dashed line provides the distribution for all MPLS tunnels, while the distribution for Opaque tunnels is plotted with a plain red line. For most non-Opaque tunnels, a large proportion (about 46%) has a most specific covering prefix length equal to 0 or 1. The fact that the two considered IP addresses are often almost entirely different is not surprising, as exiting an LSP is often equivalent to exiting an AS. Thus, the Egress of such LSPs are still within the deploying AS, while the next-IP belongs to a new domain, which is likely using a different prefix. On the contrary, a large proportion of Opaque tunnels (roughly 65%) has a most specific covering prefix greater or equal to a /30. Note that, since Juniper devices do not generate Opaque tunnels, this distribution reflects the way Cisco VPRN affects the trace's output, as mentioned in Sec. V-B. Indeed, due to those kinds of configurations, we gather the outgoing IP address of the Egress LER, followed by the incoming IP address of the next-IP. It is, thus, likely for these two addresses to share the same /30 or /31 prefix. The fact that the majority of (Egress LER, next-IP) couples share a /30 or /31 prefix is in adequacy with the fact that most Opaque tunnels seem to arise due to VPRN configurations, as can be seen in Table V

The proportion of Invisible tunnels is not negligible: 16% of tunnels in our dataset. These measurements clearly contradict our previous work suggesting that Invisible tunnels were probably 40 to 50 times less numerous than Explicit ones [8, Sec. 8]. More precisely, Invisible PHP is the most prominent configuration (87% of Invisible tunnels belongs to the Invisible PHP class), confirming so our last survey [9]. RTLA appears as being the most efficient trigger. This is partially due to

Tunnel Type	Indicator/Trigger	# LSP Revealed per Category				# LSPs	# LSRs	# LSRs per LSP
		DPR	BRPR	1Hop_Lsp	Mix			
Explicit	LSE headers	-	-	-	-	150,036	31,749	2
Implicit	qTTL	-	-	-	-	2,689	1,766	2
	Uturn	-	-	-	-	7,216	7,155	2
Opaque	LSE-TTL	22	17	43	-	3,346	52	2
Invisible UHP	Dup_Ip	1,609	1,531	686	296	4,122	862	2
Invisible PHP	Rtla	11,268	1,191	2,595	279	15,333	3,008	4
	Frpla	5,903	2,555	3,260	1,012	12,730	2,897	3
Total		18,802	5,294	6,584	1,587	195,525	47,489	3

TABLE V: Raw number of tunnels discovered by TNT per tunnel category and class (see Sec. III). No additional revelation technique is necessary for Explicit and Implicit tunnels.

the order[13] of triggers in the TNT code as it favors a high ranked trigger (Rtla) compared to low ranked one (Frpla). Moreover, Dpr works better than Brpr in practice, showing that both Juniper routers are popular for MPLS configurations and the ordered mode applied only on loopback addresses seems a common practice. It is worth noticing that in 1,784 cases (not shown in the table), Rtla was triggered but no content could be revealed. This number could represent an upper bound of Juniper VPRN that were encountered during the campaign. Those cases are not counted within the 15,333 LSPs shown in Table V. In comparison, Frpla is responsible for 11,590 unsuccessful revelation attempts. For Invisible UHP, less numerous than PHP ones ($\approx$ 2% of all LSPs), it is worth noticing (although not shown in the table) that the buddy extension was required in only 25% of the cases.

The column labeled "mix" corresponds to tunnels partially revealed thanks to Brpr and partially with Dpr. Typically, it comes from *heterogeneous MPLS clouds*. For instance, an ISP may deploy both Juniper and Cisco hardware without any homogeneous prefixes distribution. Note that it is also possible that the UHP and PHP label popping techniques co-exist when using Brpr. TNT can deal with such complex situations, making the tool robust to pitfalls encountered in the wild (5% of the Invisible tunnels encountered). The column labeled "1Hop_Lsp" corresponds to single LSR tunnels where Dpr and Brpr cannot be distinguished.

It is also worth noting that some tunnels may belong to multiple classes. We have indeed encountered situations in which an Explicit tunnel contains a few LSRs without RFC4950 enabled (i.e., being so Implicit LSR). Those tunnels and their respective LSRs are not counted in Table V and represent less than 5% of all tunnels founds.

While the column "# LSPs" provides the total amount of MPLS tunnels detected or revealed per tunnel class, the column "# LSRs" gives the contribution of each class in terms of unique IP addresses detected (with indicators) or revealed (with triggers). In both cases, the share of new MPLS data (i.e., non-explicit) that was detected (for Implicit and most Opaques) or revealed (for Invisible and some Opaques) is significant, representing more than 20% of the overall quantity of MPLS information.

Fig. 7 presents the distribution of MPLS tunnels according to their length. In 60% of the cases, LSPs contain less than

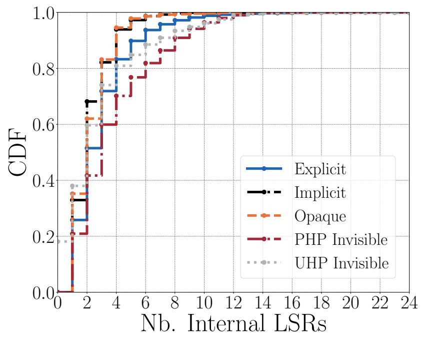


Fig. 7: Distribution of the length of the tunnels according to their classification.

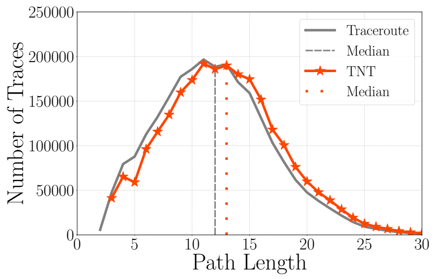


Fig. 8: Path length distribution correction with TNT. "Median" corresponds to the median path length for classic traceroute exploration (dashed grey) and when additional paths are revealed through TNT (dotted red).

4 internal LSRs, all types of tunnels combined. They are thus rather short, which is not surprising in practice, as MPLS is a technology used in transit networks, where packets are forwarded to an exit point as fast as possible in order to reduce resource consumption (hot-potato routing). This observation also confirms previous results on Invisible [9] and Explicit tunnels [32]. Besides, Opaque and Implicit LSPs seem shorter than Explicit ones, while Invisible tunnels[14] are a little longer.

Finally, Fig. 8 provides the distribution of path length with standard traceroute and with TNT. We clearly see that TNT leads to a shift of the distribution towards the right (longer paths). This shift is lower than the median length of tunnels given in the last column of Table V because all traces are taken into account, even the ones with no tunnels. Vanaubel

[13]In case several triggers apply, we prefer to use the most reliable, i.e., the less subject to any interference like BGP asymmetry.

[14]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.

et al. [9] have shown how revealing hidden tunnels also impact standard Internet model metrics.

VIII. Related Work

For years now, `traceroute` has been used as the main tool for discovering the Internet topology [1]. Multiple extensions have been provided to circumvent `traceroute` limits.

Doubletree [33, 34] has been proposed for improving the cooperation between scattered `traceroute` vantage points, reducing so the probing redundancy. Paris `traceroute` [12] and recent extensions like [35] have been developed for fixing issues related to IP load balancing. `tracebox` [5] extends `traceroute` for revealing the presence of middleboxes along a path. YARRP [36] provides techniques for speeding up the `traceroute` probing process. Reverse `traceroute` [37] is able to provide the reverse path (i.e., from the target back to the vantage point). Passenger [38] and Discarte [39] extend `traceroute` with the IP record route option. Marchetta et al. [40] have proposed to use the ICMP Parameter Problem in addition to Record Route option in `traceroute`. Finally, `tracenet` [41] mimics `traceroute` for discovering subnetworks.

`TNT` falls within the scope of the *hidden router* issue, i.e., any device that does not decrement the TTL causing the device to be invisible to `traceroute` probing. Discarte and Passenger, through the use of IP Record Route Option, allows, to some extent, to reveal hidden routers along a path. DRAGO [42] considers the ICMP Timestamp for detecting hidden routers. `TNT` goes beyond those solutions as it does not rely on specific ICMP messages or IP options. Such probes are generally filtered by operators either locally (i.e., the option/message is turned off on the router) or for transit packets (i.e., edge routers do not forward those particular packets).[15] `TNT` only relies on standard messages (`echo-request`/`echo-reply` and `time-exceeded`) that are implemented and used by the vast majority of routers and, as such, has the potential to reveal more information.

MPLS tunnels discovery has been the subject of several researches those last years. In particular, Sommers et al. [10] examined the characteristics of MPLS deployments that are explicitly identified using RFC4950 extensions, as observed in CAIDA's topology data. We proposed the first classification of MPLS tunnels [8] according to the relationship between MPLS and `traceroute`. This paper is a revision of our work in light of a deeper understanding of MPLS mechanisms, in particular for hidden tunnels (Opaque, Invisible PHP, and UHP). More recently, we have proposed techniques [9] for inferring and possibly revealing hidden tunnels: FRPLA, RTLA, BRPR, and DPR. FRPLA and RTLA were initially not used as triggers for measurements as we are doing in this paper with `TNT` (that also extends and so improves those techniques in many aspects). In previous works, they were rather used as a way to infer or validate the length of hidden tunnels. Indeed, we directed BRPR and DPR towards pre-identified high degree routers with the ITDK dataset used as an external source for triggering specific measurements (as they were suspected to be the exit point of a large number of hidden MPLS tunnels). As such, we did not provide any integrated measurement tool, on the contrary to `TNT`, a standalone active tool, with which MPLS tunnels are discovered on the fly.

[15]It has been, however, demonstrated recently that IP Record Route option might still find a suitable usage in Internet measurements if used with prudence [43].

IX. Conclusion

In this paper, we revised the MPLS classification proposed by Donnet et al. [8]. Then, we introduced `TNT` (**T**race the **N**aughty **T**unnels), an extension to Paris `traceroute` for revealing most MPLS tunnels along a path. Our fully integrated tool reveals, or at least detects, all kinds of tunnels in two simple stages. First, `TNT` relies on indicators and triggers to classify and possibly tag tunnels as hidden. Second, it launches additional probing to reveal the underlying MPLS content of false direct IP links (tagged as suspects by our set of triggers).

`TNT` provides the ability to unveil the MPLS ecosystem deployed by ISPs. Recent works have indeed shown that MPLS is largely deployed by most ISPs [8, 10, 32] for many reasons such as scalability or Traffic-Engineering. By running `TNT` periodically from largely distributed measurement platforms (e.g., Archipelago, RIPE Atlas), we expect to see numerous studies using our tool in order to correct graph properties and related models. `TNT` aims to provide a better understanding of the actual and current Internet topology.

Acknowledgments

Authors would like to thank kc claffy and her team at CAIDA for letting them deploying `TNT` on the Archipelago infrastructure. In addition, part of Mr. Vanaubel's work was supported by an internship at CAIDA, under the direction of Young Hyun.

References

[1] B. Donnet and T. Friedman, "Internet topology discovery: a survey," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 4, pp. 2–15, December 2007.

[2] 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.

[3] R. Pastor-Satorras and A. Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, 2004.

[4] 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.

[5] 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.

[6] 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.

[7] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Engineering Task Force, RFC 3031, January 2001.

[8] 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.

[9] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet, "Through the wormhole: Tracking invisible MPLS tunnels," in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2017.

[10] 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.

[11] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the Internet," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

[12] 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.

[13] L. Andersson, I. Minei, and T. Thomas, "LDP specification," Internet Engineering Task Force, RFC 5036, October 2007.

[14] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," Internet Engineering Task Force, RFC 3209, December 2001.

[15] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis, "A framework for IP based virtual private networks," Internet Engineering Task Force, RFC 2764, February 2000.

[16] Center for Applied Data Analysis, "The CAIDA UCSD internet topology data kit," March 2016, see http://www.caida.org/data/internet-topology-data-kit.

[17] k. claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *Proc. IEEE Cybersecurity Application and Technologies Conference for Homeland Security (CATCH)*, March 2009.

[18] Y. Vanaubel, J.-R. Luttringer, P. Mérindol, J.-J. Pansiot, and B. Donnet, "TNT, watch me explode: A light in the dark for revealing MPLS tunnels," in *Proc. IFIP Network Traffic Measurement and Analysis Conference (TMA)*, June 2019.

[19] 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.

[20] L. Andersson and R. Asati, "Multiprotocol label switching (MPLS) label stack entry: EXP field renamed to traffic class field," Internet Engineering Task Force, RFC 5462, February 2009.

[21] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS label stack encoding," Internet Engineering Task Force, RFC 3032, January 2001.

[22] P. Agarwal and B. Akyol, "Time-to-live (TTL) processing in multiprotocol label switching (MPLS) networks," Internet Engineering Task Force, RFC 3443, January 2003.

[23] D. Aydin, "CISCO vs. Juniper MPLS," June 2014, see http://monsterdark.com/cisco-vs-juniper-mpls/.

[24] L. De Ghein, *MPLS Fundamental: A Comprehensive Introduction to MPLS (Theory and Practice)*. CISCO Press, November 2006.

[25] R. Bonica, D. Gan, D. Tappan, and C. Pignataro, "ICMP extensions for multiprotocol label switching," Internet Engineering Task Force, RFC 4950, August 2007.

[26] T. Fiola and J. Panagos, *This Week: Deploying MPLS*, ser. Junos Networking Technologies Series. Juniper Networks Books, April 2011.

[27] Y. Vanaubel, J.-R. Luttringer, P. Mérindol, J.-J. Pansiot, and B. Donnet, "Tnt, watch me explode: A light in the dark for revealing MPLS tunnels," arXiv, cs.NI 1901.10156, February 2019.

[28] E. Rosen and Y. Rekhter, "BGP/MPLS IP virtual private networks (VPNs)," Internet Engineering Task Force, RFC 4364, February 2006.

[29] J.-F. Grailet, F. Tarissan, and B. Donnet, "TreeNET: Discovering and connecting subnets," in *Proc. Traffic Monitoring and Analysis Workshop (TMA)*, April 2016.

[30] J.-F. Grailet and B. Donnet, "Revisiting subnet inference WISE-ly," in *Proc. IFIP Network Traffic Measurementand Analysis Conference (TMA)*, June 2019.

[31] ——, "Towards a renewed alias resolution with space search reduction and IP fingerprinting," in *Proc. IFIP Network Traffic Measurementand Analysis Conference (TMA)*, June 2017.

[32] 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.

[33] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. ACM SIGMETRICS*, June 2005.

[34] 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.

[35] K. Vermeulen, S.-D. Strowes, O. Fourmaux, and T. Friedman, "Multilevel mda-lite paris traceroute." in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2018.

[36] R. Beverly, "Yarrp'ing the Internet: Randomized high-speed active topology discovery," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2016.

[37] 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, see https://www.revtr.ccs.neu.edu.

[38] R. Sherwood and N. Spring, "Touring the internet in a TCP sidecar," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2006.

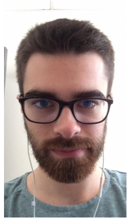
[39] R. Sherwood, A. Bender, and N. Spring, "Discarte: a disjunctive Internet cartographer," in *Proc. ACM SIGCOMM*, August 2008.

[40] 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.

[41] M. E. Tozal and K. Sarac, "TraceNET: an Internet topology data collector," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

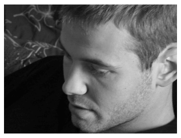
[42] P. Marchetta and A. Pescapé, "DRAGO: Detecting, quantifying and locating hidden routers in traceroute IP paths," in *Proc. Global Internet Symposium (GI)*, April 2013.

[43] B. J. Goodchild, Y.-C. Chiu, R. Hansen, H. Lua, M. Calder, M. Luckie, W. Lloyd, D. Choffnes, and E. Katz-Bassett, "The record route option is an option!" in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2017.



Jean-Romain Luttringer received his master's degree in Computer Science from the University of Strasbourg (France) in 2019, where he is currently pursuing a Ph.D. degree within the Networks Research Group of the ICube laboratory. His research interests includes routing, network measurements, path computation and Segment Routing.

Yves Vanaubel received his degree in Computer Science Engineering from the Université de Liège in 2012. He obtained his Doctoral degree in Computer Science Engineering from the same University in 2018. His research interest was Internet topology discovery, focusing on revealing hidden MPLS information. Mr. Vanaubel is now working as a Research Engineer in the Smart Grids team in the University of Liège



Pascal Mérindol received his Ph.D. degree from the University of Strasbourg (France) in 2008. Then, he spent two years in Belgium at the Université catholique de Louvain as a post-doctoral researcher. He is now Associate Professor at the Networks Research Group of the ICube laboratory in the University of Strasbourg. His main research topics are routing and Internet measurements.

Jean-Jacques Pansiot received a M.Sc. in Computer Science from Nancy University (France, 1972), a Ph.D. in Computer Science from Cornell University (USA, 1976). He joined the Department of Computer Science from the same University where he was successively appointed as Assistant Professor, Associate Professor, and Full Professor (1984). He led the Network and Protocol group of the LSIIT Laboratory and is now retired. His research interests included traffic engineering and Internet cartography.

Benoit Donnet received his Ph.D. degree in Computer Science from the Université Pierre et Marie Curie in 2006 and has been a PostDoc until 2011 at the Université catholique de Louvain (Belgium). Mr. Donnet joined the Montefiore Institute at the Université de Liège since 2011 where he was appointed successively as Assistant Professor and Associate Professor. His research interests are about Internet measurements, network modeling, middleboxes, new Internet architectures, and Computer Science Education.