# TNT: Technical Report

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

June 7, 2018

***Abstract*— Internet topology discovery has been a recurrent research topic for nearly 20 years now. Usually, it works by sending hop-limited probes (i.e., `traceroute`) towards a set of destinations to collect topological data in order to infer the Internet topology at a given scale (e.g., at the router or the AS level). However, `traceroute` comes with multiple limitations, in particular with layer-2 clouds such as MPLS that might hide their content to `traceroute` exploration. Thus, the resulting Internet topology data and models are incomplete and inaccurate.**

**In this report, we introduce `TNT` (Trace the Naughty Tunnels), an extension to Paris traceroute for revealing most (if not all) MPLS tunnels along a path. `TNT` works in two basic stages. First, along with `traceroute` probes, it looks for evidences of the potential presence of hidden tunnels. Those evidences are surprising patterns in the `traceroute` output, e.g., abrupt and significant TTL shifts. Second, if alarms are triggered due to the presence of such evidences, `TNT` launches additional and dedicated probing for possibly revealing the content of the hidden tunnel. We validate `TNT` through emulation with GNS3 and tune its parameters through a dedicated measurement campaign. We also largely deploy `TNT` on the Archipelago platform and provide a quantification of tunnels, updating so the state of the art vision of MPLS tunnels. Finally, `TNT` is fully and publicly available, as well as the collected data and scripts used for processing data.**

## 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 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 organized by operators.

However, despite the work done so far, a lot of issues still need to be fixed, specially 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]. Finally, MPLS tunnels [7]) also have an impact on topology discovery as they allow to hide internal hops [8], [9].

This report 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]. Indeed, 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 replaced by a corresponding outgoing label found in an MPLS switching table. The MPLS forwarding engine is lighter than the IP forwarding engine because finding an exact match for a label is simpler than finding the longest matching prefix for an IP address. Some MPLS tunnels may be revealed to `traceroute` because MPLS routers are able to generate ICMP `time-exceeded` message when the MPLS TTL expires and the ICMP message embeds the LSE, revealing so the presence of the tunnel [11], [8]. However the MPLS architecture supports optional mechanisms that, in effect, make MPLS tunnels invisible to `traceroute` by modifying the way the packets TTL is processed. A first attempt has been made on revealing so-called invisible [9] tunnels but this is far from being complete.

This report aims at plugging the gaps in identifying and revealing the content of MPLS tunnels. This is done by introducing `TNT` (**T**race the **N**aughty **T**unnels), an open-source extension for Paris traceroute [12] including techniques for inferring and revealing MPLS tunnels content. More precisely, this report provides four contributions:

1) we complement 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, an MPLS *revelation* is launched with the objective of revealing the tunnel content. We validate the indicators, triggers, and revelations using GNS-3, an emulator running the actual IOS of real routers in a virtualized environment.[2]. We also demonstrate, through measurements, that those techniques are efficient in terms of cost (i.e., the additional amount of probes injected is reasonable, specially compared to the quality of new data discovered) and errors (false positives and false negatives);
2) we **implement** those techniques within Scamper [13] as a Paris traceroute extension, called `TNT`, and deploy it on the Archipelago infrastructure [14]. `TNT` aims at replac-

[1] Although MPLS can also be used with IPv6 [10], in this paper we consider only IPv4

[2] See `https://gns3.com/` Note that it is also possible to emulate other router brand, e.g., Juniper, with GNS-3.

| 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`.

ing the old version of Scamper and is, thus, subject to run every day towards millions of destinations. As such, we believe `TNT` will be useful to study MPLS deployment and usage over time, increasing so our knowledge and culture on this technology;

3) we **analyze** the data collected and report a new quantification on MPLS deployment in the wild, updating so previous results [8];
4) we work in a **reproducibility** perspective. As such, all our code (`TNT`, GNS-3, data processing and analysis) as well as our collected dataset are made available.[3]

The remainder of this report is organized as follows: Sec. II provides the required technical background for this report; Sec. III introduces `TNT`, our extension to `traceroute` for revealing the content of all MPLS tunnels; Sec. IV validates `TNT` through multiple GNS3 emulations; Sec. V calibrates `TNT` parameters, while Sec. VI provides results of `TNT` deployment over the Archipelago architecture; Sec. VII position `TNT` with respect to the state of the art; finally, Sec. VIII concludes this report by symmarizing its main achievements.

## II. Background

This section discusses the technical background required for the paper. Sec. II-A explains how hardware brand can be inferred from collected TTLs. Sec. II-B to Sec. II-D are dedicated to MPLS. In particular, Sec. II-B provides the basics of MPLS labels and introduces the MPLS control plane. Sec. II-C focuses on the MPLS data plane and MPLS TTL processing. Finally, Sec. II-D explains the relationships between MPLS tunnels and `traceroute` in light of Sec. II-B and II-C.

### A. Network Fingerprinting

Vanaubel et al. [15] have presented a router fingerprinting technique that classifies networking devices based on their hardware and operating system (OS). This method infers initial TTL values used by a router when forging different kinds of packets. It then builds the 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 really interesting since the two

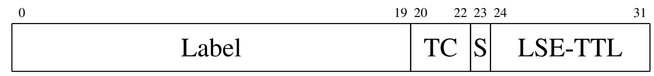



Fig. 1: The MPLS label stack entry (LSE) format.

most deployed router brands, Cisco and Juniper, have distinct MPLS behaviors and signatures.

### B. MPLS Basics and Control Plane

MPLS routers, i.e., *Label Switching Routers* (LSRs), exchange labelled 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 for forwarding the packet to the next router, a Traffic Class field for quality of service, priority, and Explicit Congestion Notification [16], a bottom of stack flag bit (to indicate whether the current LSE is the last in the stack [17])[4], and a time-to-live (LSE-TTL) field having the same purpose as the IP-TTL field [18] (i.e., avoiding routing loops).

Labels may be allocated through the *Label Distribution Protocol* (LDP) [19]. 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. With LDP, a router advertises the same label to all its neighbors for a given FEC. 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 deployed tunnels use the same routes computed by the IGP (without any interest at the first, and naive, glance) as the LFIB is built on top of the IGP FIB. Labels can also be distributed through RSVP-TE [20], when MPLS is used for Traffic Engineering (TE) purposes. In practice, most operators deploying RSVP-TE tunnels use LDP [9] as a 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 [21]), or, ($ii$), through independent LSP control (default configuration of Cisco routers [22, Chap. 4]). In the former mode, a LSR only binds a label to a prefix if this prefix is local (typically, the exit point of the LSR), 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 of its downstream LSR (to build 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, that is the Cisco default one, a LSR creates a label binding for each prefix it has in its RIB (connected or – redistributed in – IGP routes only) and distributes it to all its neighbors. This mode does not require any proposal from downstream LSR.

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

[4]To simplify the presentation we will consider only one LSE in the remainder of this paper

Consequently, a label proposal is sent to all neighbors without ensuring that the LSP is enabled up to the exit point of the tunnel. LSP setup takes less time but may lead to uncommon situation in which an LSP can end abruptly before reaching the exit point (see Sec. II-D for details.)

The last LSR towards a FEC is the *Egress Label Edge Router* (the Egress LER). 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 [17]). The previous LSR (*Penultimate Hop LSR* (PH, $P_3$ in Fig. 2) is in charge of removing the LSE to reduce the load on the Egress. In the *Ultimate Hop Popping* (UHP), the Egress LER advertises an explicit null label (label value of 0 [17]). The PH will use this 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 label, it can be the PH in case of PHP, the Egress LER in case of UHP or possibly another LSR in the case of independent LSP control.

### *C. MPLS Data Plane and TTL processing*

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

- PUSH (Sec. II-C.1). The first MPLS router, i.e., the tunnel entry point pushes one or several LSEs in the IP packet that turns into an MPLS one. The *Ingress Label Edge Router* (Ingress LER) associates the FEC of the packet to its LSP.
- SWAP (Sec. II-C.2). 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-C.3). 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 LH otherwise.

Fig. 2 illustrates the main vocabulary associated to MPLS tunnels.

*1) LSP Entry Behavior:* 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. Prior to pushing the LSE into the packet, the Ingress LER has to initialize the LSE-TTL (see Fig. 1). Two behaviors can be configured: either the Ingress LER resets 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 [18]. In the former case, the LSP is call 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 and then sent to an outgoing interface of the Ingress 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:* 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 (so the label is greater or equal than 0 including explicit null). Otherwise, it is a pop 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 a LSR may implement RFC 4950 [23] (as it should be the case in all recent OSes). If so, it means that 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., 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 to forward the packet through its IP plane [8]. Differences between Juniper and Cisco OS and configurations are discussed in detail in Sec. **??**.

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

Applying PHP comes with the advantage of reducing the load on the Egress LER, especially if it is the root of a large LSP-tree. This means that, when using PHP, the last MPLS operation (i.e., POP) is performed one hop before the Egress LER, on the EH. On the contrary, UHP 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.[5]

When leaving a tunnel, the router has to decide which TTL value (IP-TTL or LSE-TTL) to copy in the IP header. On one hand, if the Ingress LER has activated the `no-ttl-propagate` option, the EH should pick the IP-TTL of the incoming packet. On the other hand, the LSE-TTL should be selected when the `ttl-propagate` option has been activated. In order to synchronize both ends of the tunnel without any message exchange, two mechanisms might be used for selecting the IP-TTL at the EH: (*i*) applying a MIN(IP-TTL, LSE-TTL) operation (solution implemented for Cisco PHP configurations [22]) or, (*ii*), assuming 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) is the best option because it correctly supports

[5] The UHP feature does not seem to be available on Juniper routers when LSPs are set with LDP. Consequently, we consider PHP as the rule on Juniper.

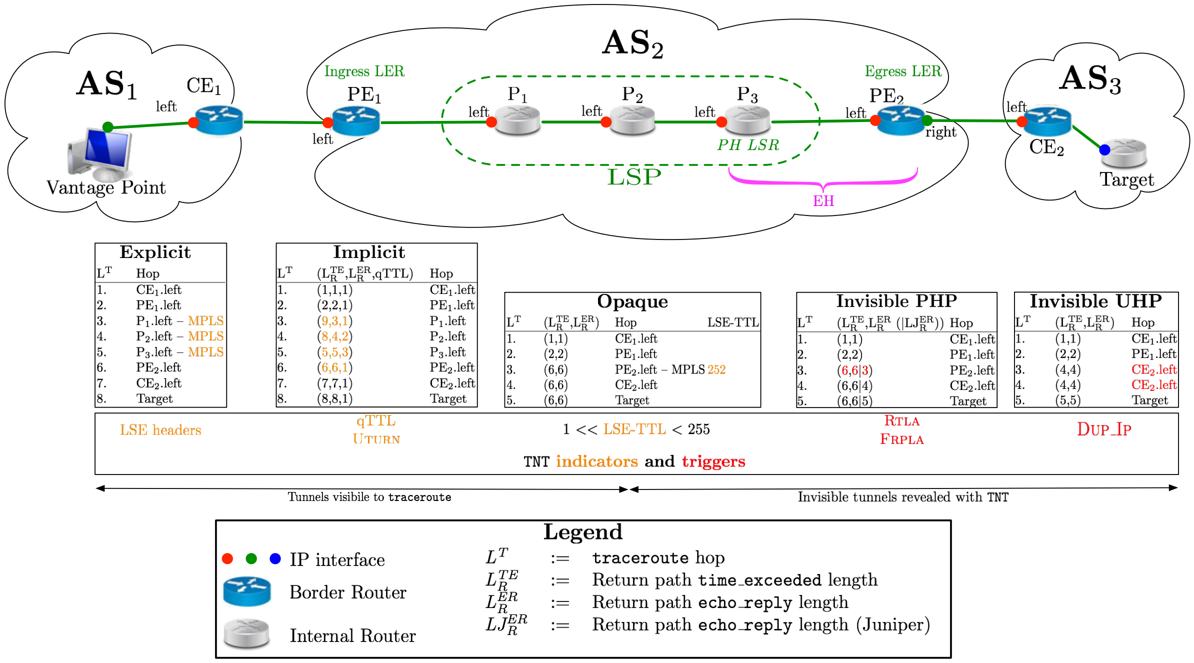



Fig. 2: Illustration of MPLS vocabulary and relationship between MPLS and `traceroute`. The figure is made of three parts. The upper part represents the network topology we use, throughout the paper to illustrate concepts. In particular, with respect to MPLS, $P_1$ is the LSP First Hop (FH), while $P_3$ is the Penultimate Hop (PH). In case of PHP, $P_3$ is the Ending Hop and is responsible for removing the LSE. In case of UHP, the LSE is removed by the Egress LER ($PE_2$). The middle part of the figure presents the MPLS Tunnel classification, as observed with `traceroute` (this classification is an update of Donnet et al. [8]). 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 (such as `time-exceeded` path length) are provided. This is used in Sec. III for illustrating `TNT`.

heterogeneous `ttl-propagate` configurations in any case while, at the same time, mitigating forwarding loop without exchanging signalization messages.

This $min$ behavior might be used for detecting the presence of hidden MPLS tunnels [9]. Indeed, it is likely that the EH generating the ICMP `time-exceeded` message will use the same MPLS cloud back to reply to the vantage point. In that case, when the reply will leave the MPLS cloud, the returning EH ($P_1$ in Fig. 2) will choose to copy the LSE-TTL in the IP-TTL, as the IP-TTL has been initialized at its maximum value on the Egress of the forward tunnel (255 for a Cisco router – see Sec. II-A). As a consequence, while the forward path hides the MPLS cloud because the min operated on the forward PH ($P_3$) will select the IP-TTL which is lower, the return path indicates its presence because the returning PH ($P_1$) will select the LSE-TTL on the contrary. In general, a sufficient condition for this pattern to occur is if the returning Ingress, which is the forward EH, re-uses the MPLS cloud back.

In practice, it is interesting to mention that this MPLS behavior is strongly dependent on the implementation and the configuration. For instance, on some Juniper OS routers (at least with JunOS Olive) or when the UHP option is activated on some Cisco IOS (at least with the 15.2 version), the MIN(IP-TTL, LSE-TTL) operation is not – systematically – applied. The EH assumes that the propagation configuration is homogeneous 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 PH (for PHP routers without MIN(IP-TTL, LSE-TTL)) or the Egress LER (for the Cisco UHP configuration) will use the IP-TTL instead of the LSE-TTL, leading so to a so-called *jump* effect with `traceroute` (i.e., as many hops as the LSP length are skipped after the tunnel). Except when implicitly 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.[6]

### *D. MPLS Tunnels Taxonomy*

According to wether LSRs implement RFC4950 or not (Sec. II-C.2) and wether they activate the `ttl-propagate` option or not (Sec. II-C.1), MPLS tunnels can be revealed to `traceroute` following Donnet et al. [8] taxonomy.

*Explicit* tunnels are those with RFC4950 and the `ttl-propagate` option activated (this is the default configuration). As such, they are fully visible by `traceroute` including labels along the LSP. *Implicit* tunnels activate the `ttl-propagate` option but not the RFC4950. No IP information is missed but LSRs are viewed as ordinary IP

[6] Those behaviors are described in Appendix IX-D.

routers, leading to a lack of "semantic" in the `traceroute` output. *Opaque* tunnels are obscured from `traceroute` as the RFC4950 is implemented but not the `ttl-propagate` option and moreover the EH that pops the last label has not received an explicit or implicit null label. Consequently, only the EH is revealed while the remainder of the tunnel is hidden. Finally, *invisible* tunnels are hidden as the `no-ttl-propagate` option is activated (RFC4950 may or not implemented).

As illustrated in Fig. 2 (middle part), explicit tunnels are the ideal case as all the MPLS information comes natively with `traceroute`. For implicit tunnels, Donnet et al. [8] have proposed techniques for identifying the tunnel based on the way LSRs process ICMP messages (see Sec. II-C.2 – the so-called UTURN) and the IP-TTL quoted in the `time-exceeded` message (the so-called qTTL) that is increased by one at each subsequent LSR of the LSP due to the `ttl-propagate` option (ICMP `time-exceeded` are generated based on the LSE-TTL while the IP-TTL of the probe is left unchanged within the LSP and, thus, quoted as such in the ICMP `time-exceeded`).

Opaque tunnels are only encountered with Cisco LSPs and are a consequence of the way labels are distributed with LDP (see Sec. II-B). Indeed, a label proposal may be sent to all neighbors without ensuring that the LSP is enabled up to the Egress LER, leading so to opaque tunnels because an LSP can end abruptly without reaching the Egress LER (where the prefix is injected in the IGP) that should bind an explicit (UHP) or implicit null label (PHP). As illustrated in Fig. 2, opaque tunnels and their length can be identified thanks to the LSE-TTL. LSPs end without a standard terminating label (implicit or explicit null) and so they *break* with the last MPLS header of the neighbor that may not be an MPLS speaker.

The `traceroute` behavior, for invisible tunnel, is different according to the way the LSE is popped from the packet (i.e., UHP or PHP), as illustrated in Fig. 2. Invisible tunnels are problematic, as they lead to a false vision of the Internet topology, creating false links, and spoiling graph metrics, such as the node degree distribution [9]. In this paper, we distinguish between invisible tunnels produced with PHP and UHP. In Donnet et al. [8], only the class "Invisible PHP" was discussed. Vanaubel et al. [9] have proposed techniques for revealing the content of invisible MPLS tunnels only in the case of PHP.

With Invisible UHP tunnels, the behavior is clearly different, at least for Cisco routers using the 15.2 IOS. Upon reception of a packet with IP-TTL of 1, the Egress LER does not decrement this TTL, but forwards the packet to the next hop ($CE_2$ in the example), so that the Egress does not show up 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. UHP indeed provokes a surprising pattern, a duplicated IP at two successive hops, illustrated as "Invisible UHP" in Fig. 2

On the contrary, PHP moves the POP function at the PH, one hop before the end of the tunnel. This PH does not decrement the IP-TTL whatever its value is. Except for some JunOS, the packet is still MPLS switched because the LSE-TTL has not expired on it. It is somehow surprising because for explicit and implicit tunnels, the PH replies on its own. It is because the LSE-TTL has also expired. In Fig. 2, we can see that there is no more asymmetry in path length for router $P_3$ proving so its reply does not follow a UTURN via the Egress. On the contrary, any other LSR on the LSP builds a `time-exceeded` message when the LSE-TTL expires and then continues to MPLS switch their reply error packet to the Egress LER unless the `mpls ip ttl-expiration pop <stack size>` command has been activated for Cisco routers. It seems to be just an option for Juniper routers with the `icmp-tunneling` command.

Note that opaque and invisible UHP are Cisco tunnels (signature $< 255, 255 >$) due to specific implementations. Invisible PHP are both Juniper (signature $< 255, 64 >$), Linux boxes (signature $< 64, 64 >$), or Cisco tunnels but they do not behave exactly the same as we will explain latter.

Sec. III extends techniques for revealing MPLS tunnels by proposing and implementing integrated measurement techniques for all tunnels (i.e., explicit, implicit, opaque, and both UHP and PHP invisible ones) in a single tool called `TNT`.

## III. `TNT`: EXPLODING MPLS TUNNELS

This section introduces our tool, `TNT` (**T**race the **N**aughty **T**unnels), able to reveal all MPLS tunnels along a path. `TNT` is an extension to Paris Traceroute [12] so that we avoid most of the problems related to load balancing. `TNT` has been implemented within scamper [13] and is freely available.[3] Sec. III-A provides an overview of `TNT`, while Sec. III-B and Sec. III-C focus on techniques for revealing hidden tunnels and how those techniques are triggered. Finally, Sec. **??** explains how we validated `TNT` on a GNS-3 platform[2], an emulator running the actual OS of real routers in a virtualized environment.

### A. Overview

Listing 1: Pseudo-code for `TNT`

```
Codes := 0, None ; 1, LSE ; 2, qTTL ; 3, UTURN ; 4, LSE-TTL;
 5, FRPLA ; 6, RTLA ; 7, DUP_IP .
trace_naughty_tunnel(target):
   prev_hop, cur_hop, next_hop = None

   for (ttl=STARTING_TTL, !halt(ttl, target), ttl++)
      state, tun_code = None
      next_hop = trace_hop(ttl)

      #first check uniform tunnel evidence with indicators
      tun_code = check_indicators(cur_hop)
      #possibly fires TNT with triggers or opaques tunnels
      if (tun_code == None)
         tun_code = check_triggers(prev_hop, cur_hop,
            next_hop)
         #check if cur_hop does not belong to a uniform LSP
         if (tun_code != None)
            #potential hidden tunnel to reveal
            state = reveal_tunnel(prev_hop, cur_hop,
                 tun_code)
      elif (tun_code == LSE-TTL)
         #potential opaque tunnel to reveal
         state = reveal_tunnel(prev_hop, cur_hop, tun_code)

      #hop by hop and tunnel display
      dump(cur_hop, tun_code, state)

      #sliding pair of IP addresses
      prev_hop = cur_hop #candidate ingress LER
      cur_hop = next_hop #candidate egress LER
```

TNT is conceptually illustrated in Listing 1. At the macroscopic scale, the `trace_naughty_tunnel()` function is a simple loop that fires probes towards each processed target. TNT consists in collecting, in a hop-by-hop fashion, intermediate IP addresses (`trace_hop()` function) between the vantage point and the target. Tracing a particular destination ends when the `halt()` function returns true: the target has been reached or a gap has been encountered (e.g., five consecutive non-responding hops, etc.). TNT uses a moving window of two hops such that, at each iteration, it considers a potential Ingress LER (i.e., `prev_hop`) and a potential Egress LER (i.e., `cur_hop`) for possibly revealing an invisible tunnel between them. Indicators allow to check if the current hop does not belong to a uniform tunnel, i.e. a visible one (see line 11).

For each couple of collected IP addresses with trace_hop, 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 necessarily requiring additional probing. Generally, indicators correspond to uniform tunnels (or to the last hop of an Opaque tunnel), and are, mostly, basic evidence of visible MPLS presence such, as LSEs quoted in the ICMP `time-exceeded` packet – see Sec. III-B for details. Triggers are mainly unsigned values suggesting the potential presence of Invisible tunnels through a large shifting in path length asymmetry – see Sec. III-B for details. When exceeding a given threshold $\mathcal{T}$, such triggers fire path revelation methods (function `reveal_tunnel()`) between the potential Ingress and Egress LERs as developed in Sec. III-C. If intermediate hops are found, they are stored in a global stack structure named `revealed_lsrs`.

`STARTING_TTL` is a parameter used to avoid tracing repeatedly the nodes close to the vantage point [24], usually `STARTING_TTL` $\in [3, 5]$.

Finally, at each loop iteration, the collected data is dumped into a warts file, the scamper file format for storing IPv4/IPv6 `traceroute` records. This job is performed by the `dump()` function. It writes potential revealed hops (available in the global stack structure `revealed_lsrs`), and any useful information, such as tags, identifying the tunnel's type and revelation method, if any.

## *B. Indicators and Triggers*

Listing 2: Pseudo-code for checking indicators

```
1  code check_indicators(hop):
2     #hop must exist
3     if (hop == None)
4        return None
5
6     if (is_mpls(hop))
7      if (T_LSE_TTL < hop.lse_ttl < 255)
8        #opaque tunnel are both indicators and triggers
9        return LSE-TTL
10     else
11       #explicit tunnel
12         return LSE
13
14    if (hop.qttl > 1)
15       #implicit tunnel
16       return qTTL
17
18    #retrieve path length from raw TTLs
19    L_R^TE = path_len(hop.ttl_te)
20    L_R^ER = path_len(hop.ttl_er)
21
22    #UTURN will be turned into RTLA for junOS signatures
23    if (|L_R^TE - L_R^ER| > T_UTURN && !signature_is_junOS(hop))
24    #implicit tunnel
25       return UTURN
26
27    return None
```

Tunnels indicators are evidence of MPLS tunnel presence and concern cases where tunnels (or parts of them) can be directly retrieved from the original `traceroute`. They are used for Explicit tunnels and uniform/visible tunnels in general. Explicit tunnels are indicated through LSEs directly quoted in the ICMP `time-exceeded` message – See line 12 in Listing 2 and `traceroute` output on Fig. 2. It is worth noting that Fig. 2 highlights the main patterns TNT looks for firing or not additional path revelation in a simple scenario where forward and return paths are symmetrical.

The indicator for Opaque tunnels consists in a single hop LSP with the quoted LSE-TTL not being equal to 1, due to the way labels are distributed within some Cisco routers (see Sec. II-B). 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 a piece of evidence in itself. It is illustrated in lines 7 to 9 in Listing 2, where a hop is tagged as Opaque if the quoted LSE-TTL is between a minimum threshold, $\mathcal{T}_{\text{LSE\_TTL}}$(see Sec. V for fixing a value for the threshold) and 254 (LSE-TTL is initialized to 255 [18]). Note that this pattern resulting from an Opaque tunnel is both an indicator and a trigger: TNT passively understands the tunnel is incomplete and try to reveal its content with new active measurements.

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. 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 14 in Listing 2. Second, the UTURN indicator 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 probing source. However, they reply directly to other kinds of probe (e.g., `echo-request`) using their own IP forwarding table, if available. As a result, in general, return paths are shorter for `time-exceeded` packets than `echo-request` messages. Thereby, UTURN is the signature related to the difference in these lengths. This is illustrated in Fig. 2 (Implicit and Explicit tunnels follow the same behavior except for RFC4950 implementation). On $P_1$, we have UTURN ($P_1$) = $L_R^{TE}$- $L_R^{ER}$ = 9 - 3 = 6. With a symmetric example, one can formalize the UTURN pattern for an LSR $P_i$ in an LSP of length $LL$ as follows:

$$\text{UTURN}(P_i) = 2 \times (LL - i + 1). \tag{1}$$

Due to the iBGP path heterogeneity (the IGP tie-break rule in particular), the BGP return path taken by the ICMP `echo-reply` message can be different from the BGP return path taken by the `time-exceeded` reply. This is illustrated in Fig. 3a where the two return paths in blue and red can differ even outside the AS ($L"_R^{TE}$ can be distinct of $L"_R^{ER}$). As a result, and because it may differ at each intermediate hop, the UTURN indicator does not necessarily follow exactly Eqn. 1. A small variation may then appear in practice. In particular, a value of 0 can hide a true Implicit hop.

For JunOS routers, the situation is quite different. It turns out that, by default (i.e., without enabling the `icmp-tunneling` feature – see Appendix IX-A.2 for details), these routers send `time-exceeded` replies directly to the source, without forwarding them to the egress LER. The UTURN indicator becomes then useless. Moreover, for routers having the JunOS signature, the UTURN indicator and the RTLA trigger are computed in the same way. Thus, to avoid any confusion, `TNT` introduces an exception for such OS signatures (line 23 in Listing 2), and first considers the difference as a trigger, and then falls back to an indicator if the revelation fails (not shown in Listing 1 for clarity). In addition, when `icmp-tunneling` is enabled, `time-exceeded` replies start with a TTL of 254, implying a bigger difference with `echo-request` replies, as it can be seen in Fig. 2: UTURN($P_1$) =LJ$^{ER}_{R}$-LJ$^{TE}_{R}$ $= 10-3 = 7$ instead of 6 if $P_1$ runs a Cisco OS.

Listing 3: Pseudo-code for checking triggers

```
1  code check_triggers(prev_hop, cur_hop, next_hop):
2     #prev_hop and cur_hop must exist
3     #duplicate IP checked on cur_hop and next_hop
4     if (prev_hop == None or cur_hop == None or prev_hop ==
         cur_hop)
5        return None
6
7     if (cur_hop == next_hop)
8        #invisible UHP tunnel
9        return DUP_IP
10    #retrieve path length from raw TTLs
11    L_R^TE = path_len(cur_hop.ttl_te)
12    L_R^ER = path_len(cur_hop.ttl_er)
13    L^T = cur_hop.probe_ttl
14
15    if (sign_is_junOS(cur_hop))
16       #for the JunOS signature
17       if (L_R^TE - L_R^ER >= T_RTLA)
18          #invisible PHP tunnel with JunOS
19          return RTLA
20    else
21       #for other signatures (raw TTLs are initialized the
          same)
22       if (L_R^TE - L^T >= T_FRPLA)
23          #invisible PHP tunnel with other known OS
24          return FRPLA
25
26    return None
```

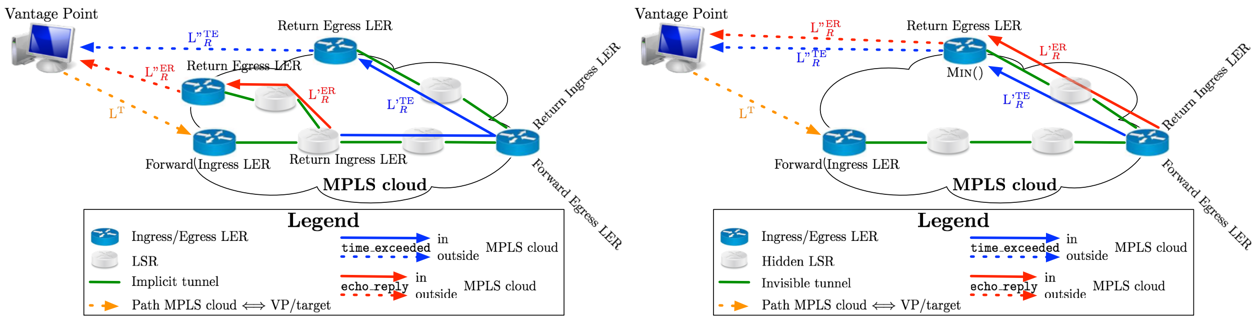
Indicators are MPLS passive evidence that can also prevent `TNT` from firing new probes (with the exception of LSE-TTL which is also a trigger for Opaque tunnels). On the contrary, triggers are active patterns suggesting the presence of invisible tunnels (both PHP and UHP) that could be revealed using additional probing (see Sec. III-C). Listing 3 provides the pseudo-code for checking triggers.

First, we look for potential Invisible UHP tunnel (line 7). As explained in Sec. II-D, Invisible UHP tunnels occur with Cisco routers using IOS 15.2. When receiving a packet with an IP-TTL of 1, the Egress LER does not decrement the TTL but, rather, forwards it directly to the next hop. Consequently, the Egress LER does not appear in the trace while, on the contrary, the next hop ($CE_2$ in Fig. 2) appears twice (duplicate IP address in the trace output).

The two remaining triggers, RTLA (Return Tunnel Length Analysis [9]) and FRPLA (Forward/Return Path Length Analysis [9]), work by using three path lengths, which are L$^{TE}_{R}$ (the `time-exceeded` path length), L$^{ER}_{R}$ (the `echo-reply` path length), and L$^{T}$ (the forward `traceroute` path length). 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 (obtained based on `traceroute` probe and reply messages). `TNT` tries to capture significative differences between these lengths to infer the presence of MPLS tunnels, relying on two common practices of LSRs, in particular the EH, developed in the previous subsection. 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 apply the MIN(IP-TTL, LSE-TTL) operation at the EH of the return tunnel. These triggers respectively infer the exact (RTLA) or approximate (FRPLA) return path length. Indeed, FRPLA is subject to BGP path asymmetry (and so, to false positives or negatives) in opposition to RTLA when it applies (it may produce some false alarms but only due to ECMP). In the absence of invisible tunnel, we expect those triggers to have a value equal or close to 0. Indeed, in such a case, we should have L'$^{ER}_{R}$ $=$ L'$^{TE}_{F}$ $=$ L'$^{TE}_{R}$ $= 1$ if BGP does not interfere (see Fig. 3). Therefore, any significant deviation from this value is interpreted as the potential presence of an Invisible MPLS cloud, and thus, brings `TNT` to trigger additional path revelation techniques (see Sec. III-C). In practice (look at Fig. 3b), we expect to have L'$^{ER}_{R}$ $=$ L'$^{TE}_{F}$ $= 1$ (due to the MIN for the `echo-reply` return tunnel and the pipe mode for the forward tunnel) while L'$^{TE}_{R}$ directly provides the actual return tunnel length (with a value $\geq 1$). It is due to the MIN operation applied by the EH of the return tunnel, which selects the LSE-TTL of the `time-exceeded` reply, and keeps the IP-TTL for the `echo-reply` packet. Indeed, in the case of the `time-exceeded` message, the return Ingress LER (i.e., the forward Egress LER) initializes the LSE-TTL with the same value as the IP-TTL, meaning 255. For `echo-reply` packets, the IP-TTL is set to 64. RTLA is not subject to any BGP asymmetry because we have L"$^{ER}_{R}$ $=$ L"$^{TE}_{R}$, i.e. BGP return paths have the same length. Indeed, the two messages use the same physical path, the only difference being the MIN operation applied at the EH of the return tunnel, if any.

To check for those triggers, we first extract the three key distances thanks to the reply IP-TTLs received by the vantage point (lines 11 to 13 in Listing 3). As explained by Vanaubel et al. [9], RTLA only works with JunOS routers, while FRPLA is more generic. Therefore, prior to estimate the triggers, `TNT` uses network fingerprinting (see Sec. II-A) to determine the router brand of the potential Egress LER (line 15 in Listing 3).

In the presence of a JunOS hardware, L$^{TE}_{R}$ is compared to L$^{ER}_{R}$, as in case of an Invisible tunnel, L$^{TE}_{R}$ is supposed to be greater than L$^{ER}_{R}$. Indeed, with this routing platform, `time-exceeded` and `echo-reply` packets have different initial TTL values (see Table I), 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 (the L$^{ER}_{R}$ appears longer than L$^{TE}_{R}$ as the MIN operation




(a) Implicit tunnels.

(b) Invisible tunnels.

Fig. 3: Indicators and triggers illustration for implicit and invisible tunnels. Notations $L'^{x}_{y}$ and $L''^{x}_{y}$ refer to a given sub-length of an ICMP packet `x` on the `y` path (`y` being the forward or return path and `x` being a `echo-reply` or `traceroute` ICMP packet, see Fig. 2). For example, $L'^{TE}_{R}$ gives the return path of the `time-exceeded` within the MPLS cloud, while $L''^{TE}_{R}$ is the return path of the `time-exceeded` between the MPLS cloud and the vantage point. Consequently, we have $L^{TE}_{R}$ = $L'^{TE}_{R}$ + $L''^{TE}_{R}$.

results in a different pick). This difference represents the number of LSRs in the return LSP, and is compared to a pre-defined threshold $\mathcal{T}_{\text{RTLA}}$(line 17 in Listing 3). This threshold (see Sec. V for the parameter calibration) filters all the LSPs shorter than the limit it defines. In the case depicted in Fig. 2: RTLA$(PE_2)$ := $L^{ER}_{R}$- $L^{TE}_{R}$ =$L'^{ER}_{R}$− $L'^{TE}_{R}$ $= 6 - 3 = 3$. Indeed, for the `echo-reply` message, we have $TTL\_IP = 64 = min(TTL\_IP = 64, TTL\_MPLS = 252)$ instead of $TTL\_IP = 252 = min(TTL\_IP = 255, TTL\_MPLS = 252)$ for the `time-exceeded` reply. Note that an invisible shadow effect also applies for RTLA after the Invisible tunnel, as the trigger will still be positive for a few nodes after the egress LER.

FRPLA is more generic and applies thus to any configuration. FRPLA allows to compare, at the AS granularity, the length distribution of forward (i.e., $L^{T}$) and return paths (i.e., $L^{TE}_{R}$). Return paths are expected to be longer than forward ones, as the tunnel hops are not counted in the forward paths 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 statistically analyze their length difference and check if a shift appears (see Line 22 in Listing 3). This is illustrated in Fig. 2 ("Invisible PHP") in which $L^{T}$ is 3 while $L^{TE}_{R}$ is equal to 6, leading so to an estimation of the return tunnel length of 3. In general, when no IP hops is hidden, we expect that the resulting distribution will look like a normal 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 makes probably use of the `no-ttl-propagate` option. In order to deal with path asymmetry, `TNT` uses a threshold, $\mathcal{T}_{\text{FRPLA}}$(see Sec. V for calibrating this parameter), greater than 0 to avoid generating too much false positives (revelation attempt with no tunnel). The MIN effect also results in an invisible shadow after the hidden LSP: FRPLA$(CE_2) = 2$ and FRPLA$(CE_3) = 1$, etc until the situation returns to normal. Note that the RTLA and FRPLA shadows are the reasons why `TNT` does not look for consecutive Invisible tunnels in a trace.

Finally, for Invisible UHP, one can observe that no MIN shift applies on the return path, as only the duplicate effect is visible.

Threshold calibration will be discussed in details in Sec. V. The optimal calibration 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, their codes, reflects their reliability, and so, their respective success rates (and their resulting states): the lower the code (i.e. the higher its priority), the more reliable (and higher the revelation success rate). Thus, if a hop matches simultaneously multiple triggers (RTLA and FRPLA for example), it is tagged with the one having the highest priority (i.e., RTLA in our example).

### *C. Hidden Tunnels Revelation*

Listing 4: Pseudo-code for revealing invisible tunnels

```
state reveal_tunnel(ingress, egress, tun_code):
  #ingress and egress hops must exist
  if (ingress == None or egress == None)
    return None
  buddy_bit = False
  #standard traceroute towards the candidate egress
  target = egress
  route = trace(REV_STARTING_TTL, target)

  if (last_hop(route) != egress)
    #the target does not respond (revelation is not
        possible)
    return TARGET_NOT_REACHABLE
  else if (ingress ∉ route)
    #the forwarding path differs (revelation is not
        possible)
    return ING_NOT_FOUND
  else if (distance(ingress, egress, route) > 1)
    #path segment revelation with DPR
    push_segment_to_revelation_stack(ingress, egress, route
        )
    return DPR
  else
    ttl = ingress.probe_ttl + 1
    revealed_ip = extract_hop(ttl, route)

    for iTR=0;;
      if (revealed_ip == target)
        if (tun_code != DUP_IP || buddy_bit)
          #no more progression in the revelation
```

```
28          break
29        else
30          #try with the buddy for the DUP_IP trigger
31          target = buddy(revealed_ip)
32          buddy_bit = True
33      else
34        #a new hop has been revealed
35        iTR++
36        push_hop_to_revelation_stack(revealed_ip)
37        target = revealed_ip
38        buddy_bit = False
39
40      revealed_ip = traceHop(ttl, target)
41
42  if (iTR == 0)
43    #no revelation (fail)
44    return NOTHING_TO_REVEAL
45  if (iTR == 1)
46    #single hop revealed LSP (DPR ≈ BRPR)
47    return 1HOP_LSP
48  else
49    #hop by hop revelation with BRPR
50    return BRPR
```

Listing 4 offers a simplified view of the TNT tunnel revelation. The first step consists in launching a standard traceroute towards the candidate Egress[7] (line 8 in Listing 4). REV_STARTING_TTL is the starting TTL used for the revelation, which corresponds to 2 hops before the candidate Ingress hop, by default. During this first attempt, TNT may fail to reach the candidate Egress (line 12), and/or the candidate Ingress (line 15) when collecting the active data. Otherwise, TNT may reveal a tunnel and four additional output states can arise:

- an LSP composed of at least 2 LSRs is revealed in the first trace towards the egress (line 19 – DPR, Direct Path Revelation [9]);
- an LSP having more than one LSR is revealed using several iterations (line 50 – BRPR, Backward Recursive Path Revelation [9]).
- nothing is revealed, the candidate Ingress and Egress are still consecutive IP addresses in the trace towards the candidate Egress (line 44);
- a single-hop LSP is revealed (line 47) although several iterations have been tried: DPR and BRPR cannot be distinguished for one hop LSPs.

With the default configuration on Cisco IOS 15.2, an additional test, called *buddy* (line 31), is required to retrieve the outgoing IP interface of the Egress LER (the right interface, in green, on $PE_2$ in Fig. 2), and so, force replies from its incoming IP interface (the left one, in red, on $PE_2$ in Fig. 2). The buddy() function assumes a point-to-point connection between the Egress LER and the next hop (IP addresses on this point-to-point link are called *buddies*). In most cases, the corresponding IP addresses belong to a /31 or a /30 prefix [4]. Note that according to the IP address submitted to buddy(), the test may require additional probing to infer the right prefix. In particular, specific UDP probing is necessary in order to provoke destination-unreachable messages. Such error messages, as time-exceeded ones, enable to get the incoming interface of the targeted router (instead of echo-reply that are indexed with the target IP).

[7]We use the term *candidate* as, at this point, we are not completely sure an MPLS tunnel is hidden there.

DPR (Direct Path Revelation) works when there is no MPLS tunneling for internal IGP prefixes other than loopback addresses, i.e., the traffic to internal IP prefixes is not MPLS encapsulated (default Juniper configuration but can also be easily configured on Cisco devices – see Sec. II-B) . With PHP, BRPR (Backward Recursive Path Revelation) works because the target ($PE_2$.left on Fig. 2) belongs to a prefix being also advertised by the PH. Thus, the probe is popped one hop before the PH ($P_3$ on Fig. 2), and it appears in the trace towards the Egress incoming IP interface, e.g., $PE_2$.left on Fig. 2. BRPR is then applied recursively on the newly discovered interface until no new IP address is revealed. BRPR works also natively with UHP on IOS 12.4 (i.e., without the buddy() function), for the same reason as for PHP: the prefix is local and shifts the end of the tunnel one hop before and, in this implementation, the EH replies directly. On the contrary, TNT needs to use the buddy() function at each step for IOS 15.2 enabling UHP, because the EH silently forwards the packet one hop ahead. Vanaubel et al. [9] provides more details on DPR and BRPR.

## IV. Reproducibility and Practical BGP Configurations

We use the GNS3 emulation environment for several purposes. First, we aim at verifying that the inference assumptions we considered in the wild are correct and reproducible in a controlled environment. Second, some of the phenomena we exploit to reveal tunnels in the wild have been directly discovered in our testbed. Indeed, using our testbed we reverse-engineered the TTL processing (considering many MPLS configurations, we study the POP operation in particular) of some common OSes used by many real routers. Finally, it is also useful for debugging TNT to test its features in this controllable environment. Generally speaking, we aim at reproducing with GNS3 all common behaviors observed in the wild, and, on the opposite, we also expect to encounter in the wild all basic behaviors (based on standard MPLS and BGP configurations) we build and setup within GNS3.

In practice, we have considered four distinct router OSes: two Cisco standard IOS (12.4 and 15.2), and two virtualized versions of JunOS (Olive and VMX, the only Juniper OS we succeeded to emulate within GNS3). We envision in a near future to also test the IOS XR and some other Juniper OSes, if possible, but we believe that our tests are already representative enough of most behaviors existing in the wild.

In our emulations, topologies (see Fig. 2) are configured as follows. We assumed that LERs are AS Provider-Edge (PE) routers, i.e., AS border routers of the ISP running (e)BGP sessions. Two main configurations are then possible to enable transit tunneling at the edges. Either the BGP next-hop can be the loopback IP address of the PE itself (with next hop self command), or it belongs to the eBGP neighbor – and in that case the connected subnet or the IP address should be redistributed in the ISP. In both cases, there exists a LDP mapping, at each Ingress LER and for any transit forwarding equivalent class (FEC) between the BGP next-hop, the IGP next-hop, and the local MPLS label to be pushed. According to the configuration at the Egress LER, when the Ingress LER is in pipe mode (see Sec. II-C.1), distinct kinds of tunnels emerge: Opaque, UHP Invisible, or PHP Invisible.

We consider the simplest possible configurations, i.e., homogeneous in terms of OS and MPLS+BGP configurations. They are consistent and symmetric MPLS configurations both in terms of signaling (LDP with the independent model using all IGP connected prefix – Cisco default mode – xor the ordered model using only loopback addresses – Juniper default mode)[8] and the propagation operation in use (pipe xor uniform)[9] at the domain scale. Using heterogeneous configurations, we discovered many intriguing corner cases that are discussed in Appendix IX. Some of them may result in incorrect TTL processing and other in hiding even more the tunnel to TNT. In some rare cases, only the Brute Force option of TNT is able to fire the path revelation that exposes tunnels.

The BGP configuration is also standard: the Egress LER enables the next-hop-self feature and so the transit traffic is tunneled via this IP address. All LSR also have a global IGP routing table thanks to a route reflector (they can answer natively to ping requests) or a redistribution in the IGP routing control plane. The AS scale BGP prefix is advertised using a global aggregation and the BGP inter-domain link is addressed by the neighbor but can be redistributed in the IGP as a connected one.

Opaque tunnels show up when enabling the neighbor <IP> ebgp-multihop <#hops> command towards the BGP neighbor whose IP address is redistributed statically in the IGP. DPR works also with Cisco IOS when enabling the mpls ldp label allocate global host-routes command. Eventually, the command mpls ldp explicit-null [for prefix-acl] allows for revealing UHP tunnels without the use of the buddy. Next paragraphs provide more practical details about the usage of such commands.

One of the most surprising behavior we observe in the wild is the one resulting from opaque tunnels. It is intriguing especially at the BGP scale because it means a badly controlled tunnel ending. It is the only kind of tunnel that requires a change in our BGP configuration to show up. Indeed, we disable the next hop self feature and select the loopback address of the neighbor as the BGP next hop using the neighbor <IP> ebgp-multihop <#hops> command to enable this possibility (IP being the address of the neighbor loopback and #hops the maximum distance expressed as a TTL value of the EBGP session). Then, we simply redistribute this IP via a static route within the IGP.

While we expect to only associate DPR to Juniper and BRPRP to Cisco configurations (as default configurations), we notice that DPR succeeds quite well in Cisco networks. It is indeed rather easy to enable such a behavior using the mpls ldp label allocate global host-routes command. It does not require complex ingoing/outgoing filtering ACL to be installed anywhere.

We also observe many different ground behaviors with UHP (only tested for LDP and so Cisco). First, when there is no duplicate IP, we sometimes collect directly null label. It appears only with the Cisco IOS 12.4. Second, we notice that the host address feature enables DPR to work without the use of the buddy. This case seems the most frequent in the wild. Even more than the default Cisco configuration that requires BRPR with the buddy. Eventually, we also discover a more sophisticated pattern in the wild: BRPR working without the use of the buddy. One can reproduce this behavior by filtering which prefixes are UHP proposed with a very simple ACL. The command mpls ldp explicit-null [for prefix-acl] should be associated with an ACL forcing the UHP proposal only for the loopback address of the Egress LER.

Using heterogeneous configurations, we discover many intriguing corner cases that are discussed in Appendix **??**. Some of them may result in incorrect TTL processing and others in hiding even more the tunnel to TNT. In some rare cases, only the Brute Force option of TNT is able to fires the path revelation that expose tunnels.

Appendix IX provides all the details of our emulations for both Cisco and Juniper configurations. All configurations were run on the topology provided by Fig. 2. The TNT running version is the one implemented in Python, available with GNS-3 scripts.[3]

[8]See Sec. II-B

[9]See Sec. II-C.1

## V. TNT Calibration and Probing Cost

Sec. III shows that TNT relies mainly on four parameters when looking for tunnels indicators or triggers: $\mathcal{T}_{\text{LSE_TTL}}$for Opaque tunnels, $\mathcal{T}_{\text{UTURN}}$for iIplicit tunnels, and $\mathcal{T}_{\text{RTLA}}$and $\mathcal{T}_{\text{FRPLA}}$for Invisible tunnels. This section aims at calibrating those parameters (Sec. V-B), as well as evaluating the probing cost associated to TNT (Sec. V-C).

### A. *Measurement Setup*

We deployed TNT on three vantage points (VPs) in the Archipelago infrastructure [14]. VPs were located in Europe (Belgium), North America (San Diego), and Asia (Tokyo).

TNT was run on April 6$^{\text{th}}$, 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.

From indicators and triggers described in Sec. III-B (see Listing 2 and 3), it is obvious that UTURN is equivalent to RTLA. Consequently, the $\mathcal{T}_{\text{UTURN}}$will have the same value than $\mathcal{T}_{\text{RTLA}}$.

For our tests, we varied $\mathcal{T}_{\text{RTLA}}$and $\mathcal{T}_{\text{FRPLA}}$between 0 and 4. A full measurement campaign was launched for each pair of parameter value (thus, a total of 25 measurement runs). Moreover for each pair, if no trigger is pulled, a so called brute force revelation is undertaken: DPR/BRPR are launched (with the use of the buddy if required). This brute force data is used as a basis to evaluate the quality and cost of each threshold value.

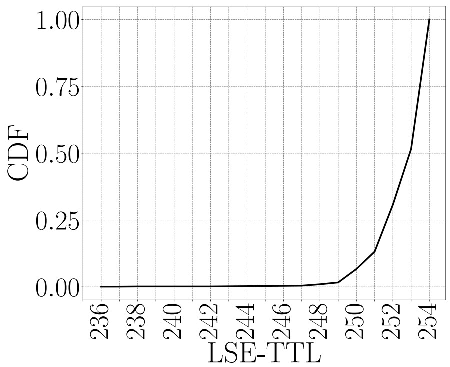



Fig. 4: Distribution of abnormal LSE-TTL values received at vantage points

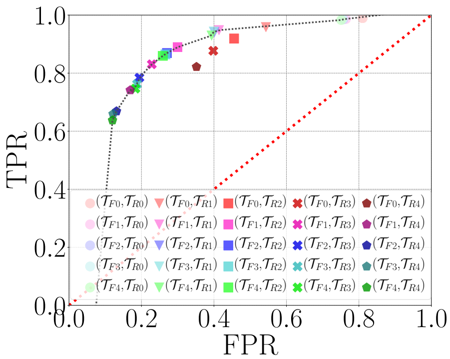



Fig. 5: Receiver operating characteristic (ROC) curve providing the efficiency of `TNT` according to values for Invisible tunnels parameters. $\mathcal{T}_{R_x}$ refers to $\mathcal{T}_{\text{RTLA}}$ with the value $x$, while $\mathcal{T}_{F_y}$ to $\mathcal{T}_{\text{FRPLA}}$ with the value $y$.

### B. Calibration

Fig. 4 provides the distribution of abnormal LSE-TTL values. By abnormal, we mean here "different from 1", which is the LSE-TTL value that should be observed in ICMP `time-exceeded` messages. Fig. 4 shows that LSE-TTL values oscillate between 236 and 254, the main proportion being located between 250 and 254. It suggests thus that, in the majority of the cases, Opaque tunnels are rather short. Consequently, a value of 236 for $\mathcal{T}_{\text{LSE\_TTL}}$ would be enough for detecting the presence of an Opaque tunnel and launching additional measurements for revealing its content.

With the help of well calibrated thresholds, the results associated to FRPLA and RTLA triggers allows for a binary classification. These triggers provide a prediction, while the results of additional probing gives the true facts when some conditions apply (see resulting states of Listing 4), i.e. being or not a tunnel. With that in mind, one can assess the performance of FRPLA and RTLA triggers 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. 5. We define TPR as the ratio of `TNT` success to the number of links being actually MPLS tunnels (having a length greater than 1): `TNT` triggers additional probing and actually reveals Invisible tunnels (we have $TPR + FNR = 1$, i.e., when adding to False Negative Rate, we obtain all links being long enough tunnels). FPR is defined as the ratio of `TNT` failure to the amount of standard IP links: it triggers for additional probing but without revealing anything (we have $FPR + TNR = 1$, i.e., when adding to True Negative Rate, we obtain all IP links without tunnels). Here, our brute force data gives the ground data that we consider reliable (i.e., revelation is fired at each hop and if nothing is revealed, we consider that there is no tunnel – we do not consider inconclusive cases where we obtain states ING_NOT_FOUND or TARGET_NOT_REACHED– see Listing 4). The ROC curve is obtained by varying the $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$ parameters between 0 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 measurement results (at the exception of $\mathcal{T}_{R_0}$ values which appear as being outliers, as expected).

We observe that the results are essentially positive for `TNT`. Some results, between ($\mathcal{T}_{R_1}$, $\mathcal{T}_{F_3}$) and ($\mathcal{T}_{R_2}$, $\mathcal{T}_{F_3}$), are even reasonably close to the perfect classification (upper left corner) and, thus, are considered as the best choice for defining our thresholds $\mathcal{T}_{\text{RTLA}}$ and $\mathcal{T}_{\text{FRPLA}}$. We expect to obtain a compromise close to 80%-20%: while we expect to reveal at least 80% of existing tunnels (MPLS links), `TNT` has a controlled overhead of 20%, i.e., it fires useless additional probing for an average limited to two actual IP links on ten.

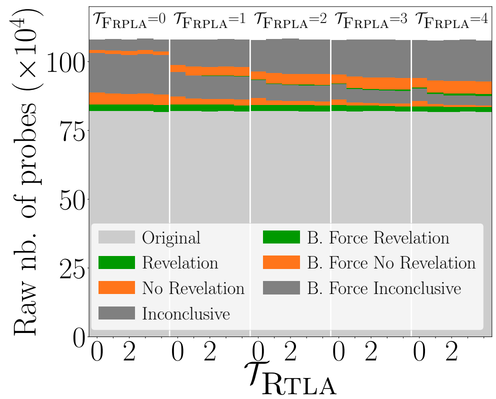



Fig. 6: Probing cost associated to `TNT` according to $\mathcal{T}_{\text{FRPLA}}$ and $\mathcal{T}_{\text{RTLA}}$ thresholds.

### C. Probing Cost

Fig. 6 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. 6) 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 – cur_hop in Listing 1 – generally due to unresponsive IP interface) or TNT was unable to reach again the candidate Ingress LER (i.e., prev_hop in Listing 1) because the destination has changed (ECMP or BGP routing noises).

If the amount of probes sent for actually revealing the content of an Invisible tunnel remains almost stable whatever the values for $\mathcal{T}_{\text{FRPLA}}$and $\mathcal{T}_{\text{RTLA}}$are, one can observe a very slow decrease meaning that there are less revealed tunnels for high values. Further, the additional traffic generated by erroneous trigger (orange) or by inconclusive revelation (dark grey) clearly decreases while $\mathcal{T}_{\text{FRPLA}}$increases. This result is aligned with Sec. V-B in which the best values for $\mathcal{T}_{\text{FRPLA}}$are between 2 and 3. Note that FRPLA is more generic but less reliable than other triggers. On the contrary, the $\mathcal{T}_{\text{RTLA}}$threshold has a minor effect on the amount of probes sent because it is more specific and more reliable.

Hatched zones (orange, dark grey, and green) correspond to the amount of probes sent using brute force. First, on the contrary to normal behavior (i.e., revelation launched according to triggers), the amount of probes sent increases with $\mathcal{T}_{\text{FRPLA}}$(the impact of $\mathcal{T}_{\text{RTLA}}$is quite negligible), as well as the amount of inconclusive revelation. Second, the amount of probes having revealed an Invisible tunnel is low compared to standard behavior.

Generally speaking, one can observe that the overhead of TNT is quite limited compared to a basic active campaign and considering the information gathered. In particular, if using correct parameters 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 for reaching a satisfying compromise where 80% of tunnels are revealed.

## VI. TNT TUNNELS QUANTIFICATION

This section aims at discussing how TNT and its features behave in the wild Internet. In particular, it analyzes the success rate of each indicator and trigger with respect to possible revelation techniques. Sec. VI-A describes the measurement setup, while Sec. VI-B discusses the results obtained.

### A. Measurement Setup

We deployed TNT on the Archipelago infrastructure [14] 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. V-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 spreads over the set of 28 vantage points to speed up the probing process.

TNT is based on Paris traceroute [12] and sends ICMP probes. A total of 522,049 distinct IP addresses (excluding traceroute targets) has been collected, with 28,350 being non publicly routable addresses (and thus excluded from our dataset). Each collected routable IP address has been pinged, only 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]

### B. Results

Table II provides the amount of probes sent by traceroute-like probing in TNT, ping, and buddy bit exploration. The row "original" refers to standard traceroute based revelation (i.e., nothing to reveal, Explicit, or Implicit tunnels).

The main results from Table II is the amount of probes involved in inconclusive revelation, split between TARGET_NOT_REACHED (TNT was unable to reach the potential Egress LER) and ING_NOT_FOUND (TNT did not cross the potential Ingress LER). In particular, TARGET_NOT_REACHED involved twice more probes than revealed tunnels. Those particular inconclusive revelations might be explained by ICMP rate limiting between the traceroute probe and additional probing (both ping and BRPR/DPR). Another explanation is that those potential Egress LERs respond to initial traceroute with an IP address that is not globally announced. As such, additional probing following the traceroute will fail as no route is available to reach them.

Table III provides the number of MPLS tunnels discovered by TNT, per tunnel type as indicated in the first column. The indicators/triggers are provided, as well as the additional revelation technique used. Without any surprise, Explicit tunnels are the most present category (76% of tunnels discovered).

Implicit tunnels represent 5% of the whole dataset, with the UTURN indicator providing more results than qTTL. However, those results must be taken with care as UTURN has been proven to be subject to false positive, while qTTL is much more reliable [25].

Opaque tunnels are less prevalent (1.7% of tunnels discovered). This is somewhat expected as Opaque tunnels are the results of particular label distribution within Cisco MPLS clouds. This confirms previous empirical results [8, Sec. 7.2]. It is also worth noticing that additional revelation techniques (DPR or BRPR) does not perform well with such tunnels (content of 98% of Opaque tunnels cannot be revealed).

The proportion of Invisible tunnels is not negligible (16% of tunnels in our dataset). Those measurements clearly contradicts 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 category), confirming so our past survey [9]. RTLA appears as being the most efficient trigger. This is partially due to the order of triggers in the TNT code because it favors high ranked trigger compared to low ranked (in case both apply). As indicated in Listing 3 (Sec. III-B), we first check for RTLA as it is proven to be more reliable than FRPLA. DPR works better than BRPR, which is obvious as it is triggered by RTLA (Juniper routers). For Invisible UHP, it

| 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 II: Raw number of probes sent by TNT over the set of 28 vantage points.

| Tunnel Type | Indicator/Trigger | Revelation Technique DPR | BRPR | 1HOP_LSP | Mix | # Tunnels |
|---|---|---|---|---|---|---|
| Explicit | LSE headers | - | - | - | - | 150,036 |
| Implicit | qTTL | - | - | - | - | 2,689 |
| | UTURN | - | - | - | - | 7,216 |
| Opaque | LSE-TTL | 22 | 17 | 43 | - | 3,346 |
| Invisible PHP | RTLA | 11,268 | 1,191 | 2,595 | 279 | 15,333 |
| | FRPLA | 5,903 | 2,555 | 3,260 | 1,012 | 12,730 |
| Invisible UHP | DUP_IP | 1,609 | 1,531 | 686 | 296 | 4,122 |
| **Total** | | 18,802 | 5,294 | 6,584 | 1,587 | 195,525 |

TABLE III: Raw number of tunnels discovered by TNT per tunnel type (see Sec. II-D). Color code for indicators/triggers is identical to Fig. 2. No additional revelation technique is necessary for Explicit and Implicit tunnels.

is worth noticing that the buddy bit, prior to BRPR or DPR revelation, was required in nearly 25% of the cases. In other cases, a simple BRPR or DPR revelation was enough to get the tunnel content. UHP seems to be often filtered for a particular FEC, e.g., only /32 host loopback addresses are advertised in LDP with UHP while other FEC are advertised with PHP (BRPR) or are not injected at all (DPR).

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, operators may deploy both Juniper and Cisco hardware without any homogeneous prefixes distribution (i.e., local prefix for Juniper, all prefixes for Cisco – See Sec. II-B for details). Note that it is also possible that the UHP and PHP label popping techniques co-exist when using our backward recurisve path revelation (BRPR). Although not explained in Sec. III for clarity reasons, TNT can deal with those more complex situations, making the tool quite robust to pitfalls encountered in the wild Internet (5% of the Invisible tunnels encountered).

Finally, the column labeled "1HOP_LSP" corresponds to one hop tunnels where DPR and BRPR cannot be distinguished. This large proportion (20%) of very short Invisible tunnels is aligned with previous works that already noticed the proportion of short Explicit tunnels [8], [11], [26].

Compared to the results presented in our previous papers ([8] in particular), we greatly improve our knowledge about MPLS and so are now able to correct our tunnel inference on many aspects. Generally speaking we had **overestimated the implicit** class while, in the same time, underestimating the use of the no-ttl-propagate option using incorrect assumptions and so extrapolation. Opaque tunnels are not due to a MPLS/IP poor interaction and concerns Cisco routers that enable the independent control mode as default (using, in addition, a specific eBGP configuration). In [8], we extrapolate the quantity of invisible tunnels considering opaque tunnels as the category gathering all LSP enabling both no TTL propagation and RFC4950. This is far from being correct as this set is actually way smaller: it consists in the intersection of no-ttl-propagate, RFC4950, the independent control model and a specific eBGP configuration where the eBGP next hop on which is based the transit traffic does not propose a normal terminating label (i.e. a null one, explicit or implicit). Our extrapolation thus clearly **underestimated invisible** tunnels because the actual opaque class is not consistent with our former classification. While invisible tunnels are much more frequent than expected because of that first mistake, it is also because implicit tunnels are less numerous than announced (and some of them turn to be badly interpreted trigger of invisible tunnels). Indeed, we realize that some inferred implicit tunnels in our previous analysis may be in reality invisible tunnels that we did not try to reveal at the time. It is because the min effect can produce the same pattern as UTURN on ICMP replies: it also provokes an asymmetry between ping and time exceeded replies as long as the downstream invisible tunnel size, we call that effect the upstream shadow of invisible tunnels. The confusion between implicit and invisible only arises for Juniper Egress but this upstream shadow also exists with Cisco routers.

## VII. 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 [24], [27] has been proposed for improving the cooperation between scattered traceroute vantage points, reducing so the probing redundancy. Paris traceroute [12] has been developed for fixing issues related to IP load balancing, avoiding so false links between IP interfaces. tracebox [5] extends traceroute for revealing the presence of middleboxes along a path. YARRP [28] provides techniques for speeding up the traceroute probing process. Reverse traceroute [29] is able to provide the reverse path (i.e., from the target back to the vantage point). Passenger [30] and Discarte [31] extend traceroute with the IP record route option. Marchetta et al. [32] have proposed to use the ICMP Parameter Problem in addition to Record Route option in traceroute. Finally, tracenet [33] mimics traceroute for discovering subnetworks.

TNT is also in the scope of the *hidden router* issue, i.e., any device that does not decrement the TTL causing the device to be transparent 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 [34] considers the ICMP Timestamp for also detecting hidden routers. TNT goes beyond those solutions as it does not rely on ICMP messages and IP option that 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).[10] 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 much more information.

## VIII. Conclusion

In this report, we introduce TNT (**T**race the **N**aughty **T**unnels is **N**ot **T**raceroute) that is an extension to Paris traceroute for revealing all MPLS tunnels along a path. As such, TNT has the potential to reveal more complete information on the exact Internet topology. We provide accurate IP level tracing functions leading so to better Internet models. For instance, it has been shown that Invisible tunnels have an impact on Internet basic graph properties [9]). Our tool reveals most kind of tunnels in two simple stages: first, it uses indicators and triggers to respectively classify and possibly tag tunnels as hidden, second it reveals the tagged tunnel content if any. TNT has the capacity to unveil the MPLS ecosystem deployed by operators. Recent works on MPLS discovery have revealed that MPLS is largely deployed by most ISP [8], [26], [11]. By running TNT on a daily (or nearly daily) basis from the Archipelago platform, we expect to see numerous researches using our tool and data to mitigate the impact of MPLS on the Internet topology. TNT has been developed with a reproducibility perspective. As such, it is freely available, as well as our dataset and scripts used for processing data.[3]

## 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.

[10] It has been, however, demonstrated recently that IP Record Route option might still find a suitable usage in Internet measurements if used with prudence [35].

[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] ——, "A brief history of MPLS usage in IPv6," in *Proc. Passive and Activement Measurement Conference (PAM)*, March/April 2016.

[11] 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.

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

[14] H. Y. claffy, kc, 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

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

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

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

[25] G. Davila Revela, 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.

[26] 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.

[27] 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.

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

[29] 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.

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

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

[32] 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.

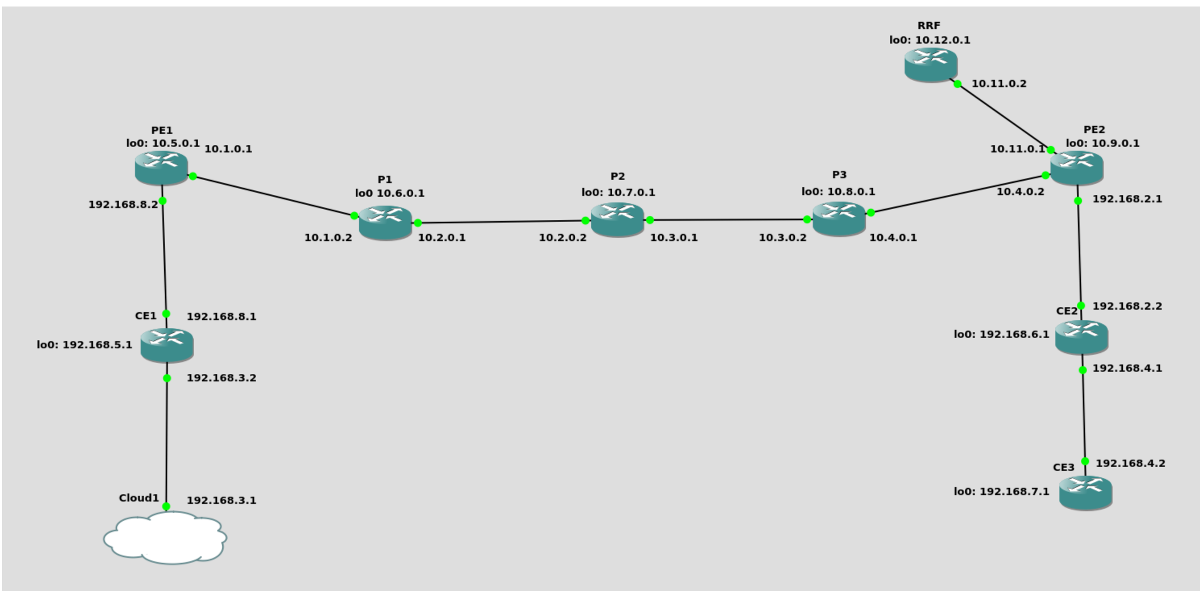
[33] M. E. Tozal and K. Sarac, "TraceNET: an Internet topology data collector," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

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

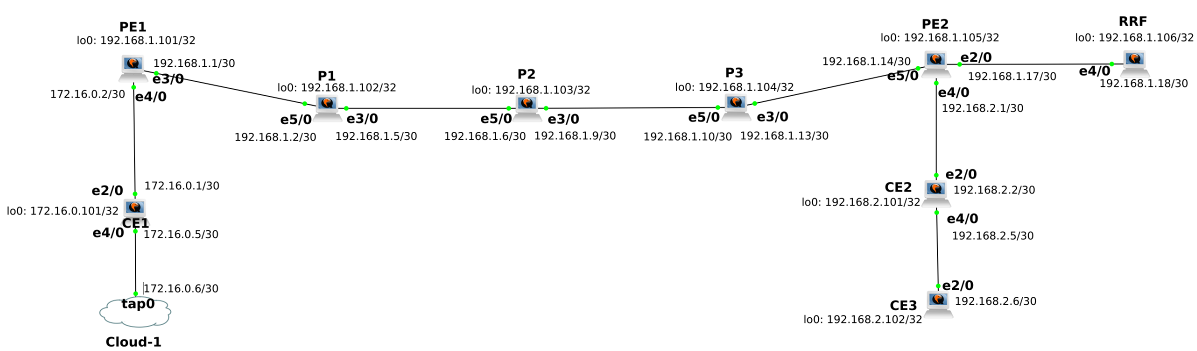
[35] B. J. Goodchild, Y.-C. Chiu, R. Hansen, H. Lua, M. Calder, M. Luckie, W. Lloyd, D. Choffnes, and E. Katz-Bassett, “The record route option is an option!” in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2017.

# IX. Appendix

This appendix illustrates the validation of TNT through GNS-3 emulations. Multiple configurations have been tested (and even more are proposed on the website[3] and can be setup using the scripts and the data online). Note that we use the version 2.1.5 of GNS3 to export the so-called portable configurations. TNT is able to deal with all those configurations (both in the wild and with the ones emulated in GNS3), making it a pretty robust tool. However, in this report we use another version of our tool to simplify the output. The output of TNT slightly differ but the conclusions are the same.




(a) Cisco topology. PE1 is the Ingress LER, PE2 the Egress LER, the LSP is set up between PE1 and the EH (P3 or PE2). The TNT target (i.e., the argument of `trace_naughty_tunnel()` function – See Listing 1) is the loopback address of CE3.




(b) Juniper topology. PE1 is the Ingress LER, PE2 the Egress LER, the LSP is set up between PE1 and the EH (P3 or PE2). The TNT target (i.e., the argument of `trace_naughty_tunnel()` function – See Listing 1) is the loopback address of CE3.

Fig. 7: Topology used for GNS-3 tests

### A. *Explicit Tunnels Validation*

We first review Explicit tunnels, i.e., tunnels with RFC4950 and `ttl-propagate` enabled (see Sec. II-D).

In the following, we distinguish Cisco (Appendix IX-A.1) and Juniper IP topologies (Appendix IX-A.2) and configurations. In particular, with Cisco configurations, PHP (LSE popped by P3) is distinguished from UHP (LSE popped by Egress LER).

For each case, we provide the configuration of routers as well as the simplified `TNT` output. Indicators and triggers (see Sec. III-B) are provided, as well as raw ICMP `time-exceeded` and ICMP `echo-reply` TTLs.

#### 1) *Cisco Explicit Configurations:*

All configurations presented here were run on the IP topology provided by Fig. 7a.

The first example provides an Explicit tunnel deployed with PHP, under Cisco IOS 15.2. The `TNT` behavior is the one expected.

**IOS 15.2 – Explicit PHP**

```
PE1
version 15.2
mpls label protocol ldp
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 15.2
mpls label protocol ldp
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 15.2
mpls label protocol ldp
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P2
version 15.2
mpls label protocol ldp
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P3
version 15.2
mpls label protocol ldp
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333
```

**`TNT` running over IOS 15.2 – Explicit PHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 27.083 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 19.895 ms
  3  left.P1 (10.1.0.2)  <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 19 | LSE-TTL : 1]  80.598 ms
  4  left.P2 (10.2.0.2)  <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 20 | LSE-TTL : 1]  69.875 ms
  5  left.P3 (10.3.0.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 20 | LSE-TTL : 1]  68.98 ms
  6  left.PE2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0] 78.17 ms
  7  left.CE2 (192.168.2.2)  <249,249> [frpla = 0][qttl = 1][uturn = 0] 78.957 ms
  8  192.168.4.2 (192.168.4.2)  <248,248> [frpla = 0][qttl = 1][uturn = 0] 110.598 ms
```

The next two configurations illustrate UHP with both IOS 12.4 and IOS 15.2. `TNT` works as expected and shows two examples of MPLS TTL processing specifically with UHP. With the 12.4 IOS, we see the null label while it is hidden with the 15.2 IOS. In addition, we can see that UHP tunnels show a UTURN signature different from PHP tunnels. This difference results from the way time-exceeded messages are handled by the LSRs. In both cases, the time-exceeded message is forwarded to the EH which replies using its own IP forwarding table. The EH changes depending on the configuration: P3 for PHP (here the EH is the PH), and PE2 for UHP (here the EH is the Egress LER). Indeed, we can see that the UTURN difference disappears at the respective EH.

**IOS 12.4 – Explicit UHP**

```

PE1
version 12.4
mpls label protocol ldp
mpls ldp explicit-null
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 12.4
mpls label protocol ldp
mpls ldp explicit-null
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 12.4
 mpls label protocol ldp
 mpls ldp explicit-null
 router bgp 3333
  neighbor 10.12.0.1 remote-as 3333


 P2
 version 12.4
 mpls label protocol ldp
 mpls ldp explicit-null
 router bgp 3333
  neighbor 10.12.0.1 remote-as 3333


 P3
 version 12.4
 mpls label protocol ldp
 mpls ldp explicit-null
 router bgp 3333
  neighbor 10.12.0.1 remote-as 3333
```

**TNT running over IOS 12.4 – Explicit UHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 22.651 ms
  2  192.168.8.2 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 230.326 ms
  3  left.P1 (10.1.0.2)  <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 22 | LSE-TTL : 1]  263.686 ms
  4  left.P2 (10.2.0.2)  <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 22 | LSE-TTL : 1]  358.238 ms
  5  left.P3 (10.3.0.2)  <249,251> [frpla = 2][qttl = 3][uturn = 2][MPLS LSE | Label : 16 | LSE-TTL : 1]  374.214 ms
  6  left.PE2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 0 | LSE-TTL : 1]  418.696 ms
  7  left.CE2 (192.168.2.2)  <249,249> [frpla = 0][qttl = 1][uturn = 0] 655.848 ms
  8  192.168.4.2 (192.168.4.2)  <248,248> [frpla = 0][qttl = 1][uturn = 0] 513.054 ms
```

**IOS 15.2 – Explicit UHP**

```

PE1
version 15.2
mpls label protocol ldp
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
mpls ldp explicit-null
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P2
version 15.2
mpls label protocol ldp
mpls ldp explicit-null
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P3
version 15.2
mpls label protocol ldp
mpls ldp explicit-null
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333
```

**TNT running over IOS 15.2 – Explicit UHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.64 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 39.87 ms
  3  left.P1 (10.1.0.2)  <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 19 | LSE-TTL : 1]  100.632 ms
  4  left.P2 (10.2.0.2)  <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 20 | LSE-TTL : 1]  80.453 ms
  5  left.P3 (10.3.0.2)  <249,251> [frpla = 2][qttl = 3][uturn = 2][MPLS LSE | Label : 20 | LSE-TTL : 1]  100.815 ms
  6  left.PE2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0] 109.089 ms
  7  left.CE2 (192.168.2.2)  <249,249> [frpla = 0][qttl = 1][uturn = 0] 98.817 ms
  8  192.168.4.2 (192.168.4.2)  <248,248> [frpla = 0][qttl = 1][uturn = 0] 119.842 ms
```

*2) Juniper Explicit Configurations:* All configurations presented here were run on the topology provided by Fig. 7b.

For Explicit tunnels, Juniper Olive and VMX behave the same. We first provide the configuration and TNT output for Explicit tunnels without UTURN effect.

**VMX – Explicit PHP (default configuration)**

```
PE1
propagate ttl

PE2
propagate ttl



P1
propagate ttl

P2
propagate ttl

P3
propagate ttl
```

**TNT running over VMX - Explicit PHP (default configuration)**

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 ( 172.16.0.5)  <255,64> [frpla = 0][qttl = 1][uturn = 0] 2.682 ms
  2  PE1 ( 172.16.0.2)  <254,63> [frpla = 0][qttl = 1][uturn = 0] 4.603 ms
  3  left.P1 (192.168.1.2)  <253,62> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1]  6.362 ms
  4  left.P2 (192.168.1.6)  <252,61> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1]  8.451 ms
  5  left.P3 (192.168.1.10)  <251,60> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1]  8.557 ms
  6  left.PE2 (192.168.1.14)  <250,59> [frpla = 0][qttl = 1][uturn = 0] 8.285 ms
  7  CE2 (192.168.2.2)  <249,58> [frpla = 0][qttl = 1][uturn = 0] 8.09 ms
  8  CE3 (192.168.2.102)  <248,57> [frpla = 0][qttl = 1][uturn = 0] 8.142 ms
```

On the contrary to Cisco configuration, Juniper does not exhibit the UTURN effect. When the LSE-TTL of a packet expires, the LSR does not send the ICMP `time-exceeded` to the EH which then forwards the packets on its own to the probing source, it replies the same with respect to other probes (e.g., `echo-request`) using its own IP forwarding table if available – resulting in general in a shorter return path (see Sec. III-B). The configuration must be explicitly stated with the `icmp-tunneling` as provided below.

**VMX – Explicit PHP (`icmp-tunneling` configuration)**

```
PE1
propagate ttl
icmp-tunneling

PE2
propagate ttl
icmp-tunneling

P1
propagate ttl
icmp-tunneling

P2
propagate ttl
icmp-tunneling

P3
propagate ttl
icmp-tunneling
```

**`TNT` running over VMX – Explicit PHP (`icmp-tunneling` configuration)**

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 ( 172.16.0.5)  <255,64> [frpla = 0][qttl = 1][uturn = 0] 2.034 ms
  2  PE1 ( 172.16.0.2)  <254,63> [frpla = 0][qttl = 1][uturn = 0] 4.646 ms
  3  left.P1 (192.168.1.2)  <246,62> [frpla = 7][qttl = 1][uturn = 7][MPLS LSE | Label : 299824 | LSE-TTL : 1]  11.424 ms
  4  left.P2 (192.168.1.6)  <247,61> [frpla = 5][qttl = 1][uturn = 5][MPLS LSE | Label : 299824 | LSE-TTL : 1]  7.994 ms
  5  left.P3 (192.168.1.10)  <251,60> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1]  6.252 ms
  6  left.PE2 (192.168.1.14)  <250,59> [frpla = 0][qttl = 1][uturn = 0] 8.585 ms
  7  CE2 (192.168.2.2)  <249,58> [frpla = 0][qttl = 1][uturn = 0] 9.369 ms
  8  CE3 (192.168.2.102)  <248,57> [frpla = 0][qttl = 1][uturn = 0] 9.232 ms
```

## *B. Opaque Tunnels Validation (Cisco only)*

Opaque tunnels only occur with Cisco routers, in some particular configuration (see Sec. II-D for details). The topology used for GNS-3 emulation is the one provided by Fig. 7a. We only show tests for IOS 15.2 as the situation is the same with IOS 12.4. In our example, we were able to reveal the content of the Opaque tunnel through BRPR, on the contrary to in the wild `TNT` deployment where Opaque tunnels revelation did not work that much (see Sec. VI). We see thus here a difference between theory and practice.

**IOS 15.2 – Opaque PHP**

```
PE1
version 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 192.168.8.1 remote-as 1024

PE2
version 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 192.168.6.1 remote-as 2048
 neighbor 192.168.6.1 ebgp-multihop 2

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


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


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

**`TNT` running over IOS 15.2 – Opaque PHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 25.164 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 40.06 ms

    OPAQUE | Length estimation : 3 | Revealed : 3 (difference : 0)
     2.1 [REVEALED] left.P1 (10.1.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 40.008 ms  - step 2
     2.2 [REVEALED] left.P2 (10.2.0.2)  <252,252> [frpla = 0][qttl = 1][uturn = 0] 40.058 ms  - step 1
     2.3 [REVEALED] left.P3 (10.3.0.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0] 90.301 ms  - step 0

  3  left.PE2 (10.4.0.2)  <250,250> [frpla = 3][qttl = 1][uturn = 0][MPLS LSE | Label : 16 | LSE-TTL : 252]  110.408 ms
  4  left.CE2 (192.168.2.2)  <250,250> [frpla = 2][qttl = 1][uturn = 0] 80.195 ms
  5  192.168.4.2 (192.168.4.2)  <250,250> [frpla = 1][qttl = 1][uturn = 0] 132.331 ms
```

## C. Invisible Tunnels Validation

This section discusses Invisible tunnels, i.e., tunnels with the `no-ttl-propagate` option enabled (see Sec. II-D).

We do a distinction between Cisco (Appendix IX-C.1) and Juniper configurations (Appendix IX-C.2). PHP (LSE popped by P3) is also distinguished from UHP (LSE popped by Egress LER).

For each case, we provide the configuration of routers as well as the `TNT` output. Indicators and triggers (see Sec. III-B) are provided, as well as ICMP `time-exceeded` and ICMP `echo-reply` TTLs.

*1) Invisible Cisco Configurations:* All configurations presented here were run on the topology provided by Fig. 7a.

**IOS 15.2 – Invisible PHP**

```

PE1
version 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 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 15.2
mpls label protocol ldp
no  propagate-ttl
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


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


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

**`TNT` running over IOS 15.2 – Invisible PHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.52 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.927 ms

    FRPLA | Length estimation : 3 | Revealed : 3 (difference : 0)
     2.1 [REVEALED] left.P1 (10.1.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 50.051 ms  - step 2
     2.2 [REVEALED] left.P2 (10.2.0.2)  <252,252> [frpla = 0][qttl = 1][uturn = 0] 60.102 ms  - step 1
     2.3 [REVEALED] left.P3 (10.3.0.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0] 59.876 ms  - step 0

  3  left.PE2 (10.4.0.2)  <250,250> [frpla = 3][qttl = 1][uturn = 0] 80.38 ms
  4  left.CE2 (192.168.2.2)  <250,250> [frpla = 2][qttl = 1][uturn = 0] 69.89 ms
  5  192.168.4.2 (192.168.4.2)  <250,250> [frpla = 1][qttl = 1][uturn = 0] 99.833 ms
```

The configuration for running standard Cisco Invisible UHP tunnels is provided below. Such a configuration might be revealed through BRPR thanks to the DUP_IP trigger.

**IOS 15.2 – Invisible UHP**

```
PE1
version 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


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


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

**TNT running over IOS 15.2 – Invisible UHP**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 3.157 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.92 ms

    Duplicate IP (Egress : 192.168.2.2) | Length estimation : 1 | Revealed : 4 (difference : 3)
     2.1 [REVEALED] left.P1 (10.1.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 50.043 ms  - step 4  (Buddy used)
     2.2 [REVEALED] left.P2 (10.2.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 49.778 ms  - step 3  (Buddy used)
     2.3 [REVEALED] left.P3 (10.3.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 69.834 ms  - step 2  (Buddy used)
     2.4 [REVEALED] left.PE2 (10.4.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 80.594 ms  - step 1  (Buddy used)

  3  left.CE2 (192.168.2.2)  <252,252> [frpla = 1][qttl = 1][uturn = 0] 80.08 ms
  4  left.CE2 (192.168.2.2)  <252,252> [frpla = 0][qttl = 1][uturn = 0] 89.891 ms
  5  192.168.4.2 (192.168.4.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0] 107.579 ms
```

With Cisco routers, it is possible to mimic an Invisible UHP tunnel with a Juniper per loopback configuration (i.e., by filtering addresses to /32 border prefixes), meaning that the tunnel content might be revealed through DPR, thanks to the DUP_IP trigger. Such a configuration is achieved with the `allocate global host-routes` command.

**IOS 15.2 – Invisible UHP (`allocate global host route` configuration)**

```
PE1
version 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
mpls ldp label
 allocate global host-routes
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
mpls ldp label
 allocate global host-routes
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
mpls ldp label
 allocate global host-routes
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P2
version 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
mpls ldp label
 allocate global host-routes
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


P3
version 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
mpls ldp label
 allocate global host-routes
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333
```

**`TNT` running over IOS 15.2 – Invisible UHP (`allocate global host route` configuration)**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 8.091 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 39.867 ms

    Duplicate IP (Egress : 10.1.0.2) | Length estimation : 1 | Revealed : 4 (difference : 3)
     2.1 [REVEALED] left.P1 (10.1.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 39.788 ms  - step 2
     2.2 [REVEALED] left.P2 (10.2.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 49.573 ms  - step 2
     2.3 [REVEALED] left.P3 (10.3.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 70.094 ms  - step 2
     2.4 [REVEALED] left.PE2 (10.4.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 89.171 ms  - step 1  ( Buddy used )

  3  left.CE2 (192.168.2.2)  <252,252> [frpla = 1][qttl = 1][uturn = 0] 120.546 ms
  4  left.CE2 (192.168.2.2)  <252,252> [frpla = 0][qttl = 1][uturn = 0] 89.892 ms
  5  192.168.4.2 (192.168.4.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0] 117.301 ms
```

It is also possible to build Invisible UHP tunnel in which the buddy mechanism is not necessary (as we discover in the wild). Simply running BRPR will make the tunnel content visible. This configuration might be achieved with the `ip access-list` command to enable Ultimate Hop Popping for external destinations only:

**IOS 15.2 – Invisible UHP (`mpls ldp explicit-null [for prefix-acl]` configuration)**

```
PE1
version 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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null for BRPR-wo-buddy
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
ip access-list standard BRPR-wo-buddy
 permit 10.9.0.1
 deny any
```

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

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

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

**TNT running over IOS 15.2 – Invisible UHP (`mpls ldp explicit-null [for prefix-acl]` configuration)**

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  192.168.3.2 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.299 ms
  2  192.168.8.2 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 14.921 ms

    Duplicate IP (Egress : 10.4.0.2) | Length estimation : 3 | Revealed : 4 (difference : 1)
     2.1 [REVEALED] 10.1.0.2 (10.1.0.2)  <253,253> [frpla = 0][qttl = 1][uturn = 0] 36.443 ms  - step 3
     2.2 [REVEALED] 10.2.0.2 (10.2.0.2)  <252,252> [frpla = 0][qttl = 1][uturn = 0] 35.879 ms  - step 2
     2.3 [REVEALED] 10.3.0.2 (10.3.0.2)  <251,251> [frpla = 0][qttl = 1][uturn = 0] 66.288 ms  - step 1
     2.4 [REVEALED] 10.4.0.2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0] 64.19 ms  - step 0

  3  CE2 (192.168.2.2)  <250,250> [frpla = 3][qttl = 1][uturn = 0] 116.643 ms
  4  CE2 (192.168.2.2)  <250,250> [frpla = 2][qttl = 1][uturn = 0] 99.93 ms
  5  192.168.4.2 (192.168.4.2)  <250,250> [frpla = 1][qttl = 1][uturn = 0] 94.185 ms
```

*2) Juniper Invisible Configurations:* All configurations presented here were run on the topology provided by Fig. 7b.

Juniper, with Olive OS, does not apply the MIN(IP-TTL, LSE-TTL) at the exit of the MPLS cloud. As such, the FRPLA trigger does not provide the return tunnel length but is equal to 1 because the ingress LER process the incoming IP TTL in a distinct way with respect to the origin of the packet (locally generated or not). Invisible PHP tunnel can, then, be revealed through DPR. Juniper LSR can be configured as followed:

**JunOS Olive – Invisible PHP**

```
PE1
no-propagate-ttl


PE2
no-propagate-ttl
```

```
P1
no-propagate-ttl

P2
no-propagate-ttl

P3
no-propagate-ttl
```

**TNT running over JunOS Olive – Invisible PHP**

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 ( 172.16.0.5)  <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.638 ms
  2  PE1 ( 172.16.0.2)  <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.898 ms

    FRPLA | Length estimation : 1 | Revealed : 3 (difference : 2)
     2.1 [REVEALED] left.P1 (192.168.1.2)  <253,62> [frpla = 0][qttl = 1][uturn = 0] 3.039 ms  - step 0
     2.2 [REVEALED] left.P2 (192.168.1.6)  <252,61> [frpla = 0][qttl = 1][uturn = 0] 3.951 ms  - step 0
     2.3 [REVEALED] left.P3 (192.168.1.10)  <252,61> [frpla = 0][qttl = 1][uturn = 0] 4.906 ms  - step 0

  3  left.PE2 (192.168.1.14)  <252,61> [frpla = 1][qttl = 1][uturn = 0] 7.043 ms
  4  CE2 (192.168.2.2)  <252,61> [frpla = 0][qttl = 1][uturn = 0] 6.891 ms
  5  CE3 (192.168.2.102)  <251,60> [frpla = 0][qttl = 1][uturn = 0] 8.978 ms
```

On the contrary to Olive, VMX applies the MIN(IP-TTL, LSE-TTL) function. As such, the behavior observed is the theoretical one. It is worth noting that configuring Juniper VMX for Invisible MPLS tunnels is identical than with Olive. Invisible tunnels are, now, revealed through DPR, with the RTLA trigger.

**JunOS VMX – Invisible PHP**

```

PE1
no-propagate-ttl


PE2
no-propagate-ttl

```

```

P1
no-propagate-ttl

P2
no-propagate-ttl

P3
no-propagate-ttl
```

**TNT running over JunOS VMX – Invisible PHP**

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 ( 172.16.0.5)  <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.96 ms
  2  PE1 ( 172.16.0.2)  <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.66 ms

   RTLA | Length estimation : 3 | Revealed : 3 (difference : 0)
     2.1 [REVEALED] left.P1 (192.168.1.2)  <253,62> [frpla = 0][qttl = 1][uturn = 0] 8.8 ms  - step 0
     2.2 [REVEALED] left.P2 (192.168.1.6)  <252,62> [frpla = 0][qttl = 1][uturn = 0] 2.134 ms  - step 0
     2.3 [REVEALED] left.P3 (192.168.1.10)  <251,62> [frpla = 0][qttl = 1][uturn = 0] 3.352 ms  - step 0

  3  left.PE2 (192.168.1.14)  <250,62> [frpla = 3][rtl = 3(3)][qttl = 1][uturn = 3] 4.569 ms
  4  CE2 (192.168.2.2)  <250,61> [frpla = 2][rtl = 2(-1)][qttl = 1][uturn = 2] 4.625 ms
  5  CE3 (192.168.2.102)  <250,60> [frpla = 1][rtl = 1(-1)][qttl = 1][uturn = 1] 4.355 ms
```

### D. Corner Cases: heterogeneous propagation configuration

This section discusses corner cases, i.e., unlikely configurations that may arise when MPLS is not homogeneously configured throughout the tunnel. TNT, like traceroute, cannot deal with those situations, but these abnormal shiftings have not been clearly encountered in practice.

*1) Cisco Jumpy Configurations:* The following Cisco configuration (for IOS 15.2) is supposed to build an UHP Invisible tunnel. However, on the contrary to the configuration provided in Appendix IX-C.1, the management of LSE-TTL is heterogeneous over the tunnel. Indeed, in this case, the Ingress LER is not configured with the no-ttl-propagate (on the contrary to the Egress LER and other routers in the tunnel). As such, the MIN(IP-TTL, LSE-TTL) operation is not – systematically – applied on the Egress while it is expected to be from the Ingress. The EH assumes that the propagation configuration is homogeneous among LERs, which is not the case here. Therefore, the Egress LER will use the IP-TTL instead of the LSE-TTL when popping the LSE. As consequence, and as shown by the TNT output, we observe that

1) the MPLS tunnel is actually Explicit;
2) a number of hops equal to the tunnel length after the MPLS tunnel are missing (here, only CE2 is missing as the platform is too short – see Fig. 7a for the Cisco topology we use), leading to a so-called *jump* effect.

We call such a configuration *Explicit Jump* and it can be observed in the qTTL of the last hop (2 instead of one plus the skipped hop).

**IOS 15.2 – Explicit Jump (heterogeneous configuration)**

```

PE1
version 15.2
mpls label protocol ldp
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
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 15.2
mpls label protocol ldp
no  propagate-ttl
mpls ldp explicit-null
router bgp 3333
 neighbor 10.12.0.1 remote-as 3333


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


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

TNT running over IOS 15.2 – Explicit Jump (heterogeneous configuration)

```
Launching TNT: 192.168.7.1 (192.168.7.1)

  1  left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 8.407 ms
  2  left.PE1 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.477 ms
  3  left.P1 (10.1.0.2)  <250,253> [frpla = 3][qttl = 1][uturn = 3][MPLS LSE | Label : 19 | LSE-TTL : 1]  79.929 ms
  4  left.P2 (10.2.0.2)  <250,252> [frpla = 2][qttl = 2][uturn = 2][MPLS LSE | Label : 20 | LSE-TTL : 1]  80.573 ms
  5  left.P3 (10.3.0.2)  <250,251> [frpla = 1][qttl = 3][uturn = 1][MPLS LSE | Label : 20 | LSE-TTL : 1]  109.577 ms
  6  left.PE2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0] 79.766 ms
  7  192.168.4.2 (192.168.4.2)  <250,250> [frpla = -1][qttl = 2][uturn = 0] 109.357 ms
```

*2) Juniper Jumpy Configurations:* In the fashion of Cisco, Juniper with the Olive OS (this is not possible with VMX) allows to configure an Explicit Jump tunnel with PHP. The configuration provided below shows such an MPLS tunnel. The EH is configured with the `no-ttl-propagate` option, while other routers are configured with `ttl-propagate`. As such, P3 will not apply the MIN(IP-TTL, LSE-TTL) when popping the label, leading so to a jump effect that is nearly as long as the tunnel itself (the Egress LER and CE2 are missing plus the qTTl at 2 on the last hop).

Olive – Explicit Jump (heterogeneous configuration)

```
PE1
propagate ttl

PE2
propagate ttl

P1
propagate ttl

P2
propagate-ttl

P3
no-propagate-ttl
```

TNT running over Olive – Explicit (heterogeneous configuration)

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 (172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.622 ms
  2  PE1 (172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.749 ms
  3  left.P1 (192.168.1.2) <253,62> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1] 2.799 ms
  4  left.P2 (192.168.1.6) <252,252> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1] 3.725 ms
  5  left.P3 (192.168.1.10) <251,251> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299776 | LSE-TTL : 1] 7.784 ms
  6  CE3 (192.168.2.102)  <248,57> [frpla = 2][qttl = 2][uturn = 0] 8.884 ms
```

The last configuration is Juniper Olive with an *Invisible Jump* configuration. This is somewhat equivalent to the Explicit Jump but for Invisible tunnels. In that case, when P3 (PHP is configured) will pop the LSE, it will not apply the MIN(IP-TTL, LSE-TTL). As a result, `TNT` will see the Ingress LER (PE1) and several hops after P3 will be missed (Egress LER and CE2). The tunnel is invisible and triggers do not work. One can notice a qTTL of 250 on the last hop of our platform: it means that `traceroute` can miss an entire path of 255 minus the length of the tunnel!

Olive – Invisible Jump configuration (heterogeneous configuration)

```
PE1
no-propagate ttl


PE2
propagate ttl

P1
no-propagate ttl

P2
no-propagate-ttl

P3
propagate-ttl
```

TNT running over Olive – Invisible Jump (heterogeneous configuration)

```
Launching TNT: 192.168.2.102 (192.168.2.102)

  1  CE1 ( 172.16.0.5)  <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.515 ms
  2  PE1 ( 172.16.0.2)  <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.712 ms
  3  CE3 (192.168.2.102)  <251,60> [frpla = 2][qttl = 250][uturn = 0] 8.553 ms
```