# TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels

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

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

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

## I. Introduction

For now twenty years, the Internet topology discovery has attracted a lot of attention from the research community [1]. First, numerous tools [2], [3], [4] have been proposed to better capture the Internet at the IP interface level (mainly based on `traceroute`) and at the router level (by aggregating IP interfaces of a router through *alias resolution* [5]). Second, the data collected has been used to model the Internet [6], 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, especially in data collection processes based on `traceroute`. For instance, collecting data about Layer-2 devices connecting routers is still an open question, although it has been addressed previously with a, nowadays, deprecated tool (i.e., IGMP-based probing) [7]. Another example is the relationship between traditional network hardware and the so-called middleboxes [8], [9]. Finally, MPLS tunnels [10] also have an impact on topology discovery as they allow to hide internal hops [11], [12]. Efforts have been made in discovering MPLS infrastructures, based on `traceroute` [11], [13], IP Record Route option [14], [15], or ICMP timestamp [16].

This paper focuses on the interaction between `traceroute` and MPLS. In a nutshell, MPLS simplifies and extends the forwarding data plane thanks to the insertion of *labels* (called *Label Stack Entries*, or LSE) before the IP header. MPLS packets are forwarded using an exact match lookup of a 20-bit value carried within the LSE. At each MPLS hop, the label of a packet is either pushed, popped, or swapped with its associated outgoing label provided within the MPLS switching table. MPLS comes with several advantages: enabling traffic engineering, providing new services such as VPRN, and ensuring inter-domain routing scalability at the AS scale. Some MPLS tunnels may be natively revealed to `traceroute` because, when the MPLS TTL expires, MPLS routers generate ICMP `time-exceeded` messages embedding the LSE [13], [11]. However, MPLS supports optional features that make tunnels more or less invisible to `traceroute`. Such features modify the way routers process the IP and MPLS TTL of a packet. By carefully analyzing several MPLS related patterns based on TTL values (e.g., the quoted forward TTL or the returned TTL of both error and standard replies), one can identify and possibly discover L3-hops hidden within an MPLS cloud. A first attempt has been already proposed for revealing so-called Invisible tunnels [12].

This paper aims at improving the efficiency of their discovery in order to reveal (or at least identify) more invisible tunnels at a lower cost. This is done by introducing `TNT` (**T**race the **N**aughty **T**unnels), an open-source scamper [18] plugin extension based on Paris traceroute [17], that includes techniques for inferring, classifying, and possibly revealing MPLS tunnels content. Compared to our previous work [11], [12], this paper provides multiple additional contributions. First, we revise our initial MPLS tunnel taxonomy by clearly distinguishing "Invisible PHP" and "Invisible UHP" tunnels and by better classifying and understanding "Opaque" tunnels. Second, we complement state of the art measurement techniques in order to reveal most MPLS tunnels and at least detect them all, even those built to hide their content. Those measurement techniques are performed on-the-fly with `traceroute` according to *indicators* and *triggers* that are used to determine the potential presence of a tunnel and possibly its nature. Third, we have implemented those techniques in `TNT` and deployed it on the Archipelago platform [19]. We verified and improved the design of `TNT` using comprehensive[1] GNS3 emulation tests. This paper also addresses the question of the calibration of `TNT` regarding its triggers. Moreover, using a large-scale

[1]Emulations details are provided in a dedicated technical report [20].

978-3-903176-17-1 / © 2019 IFIP

measurement campaign, we correct and, so, update previous results [11] that erroneously underestimated or overestimated the prevalence of some tunnel classes. Finally, all our code (`TNT`, GNS-3 configurations, data processing and analysis), as well as our collected dataset, are made available.[2]

The remainder of this paper is organized as follows: Sec. II provides the required technical background for this paper; Sec. III revises the MPLS taxonomy initially introduced by Donnet et al. [11] in the light of newly understood MPLS behaviors; Sec. IV discusses how the content of invisible tunnels might be exposed to `traceroute`; Sec. V introduces `TNT`, our extension to `traceroute` for revealing the content of all MPLS tunnels; Sec. VI discusses `TNT` parameters and its calibration, while Sec. VII presents results of the `TNT` deployment over the Archipelago architecture; finally, Sec. VIII concludes this paper by summarizing its main achievements.

## II. MPLS Background

### A. MPLS Basics and Control Plane

MPLS routers, i.e., *Label Switching Routers* (LSRs), exchange labeled packets over *Label Switched Paths* (LSPs). In practice, those packets are tagged with one or more *Label Stack Entries* (LSE) inserted between the frame header and the IP header. Each LSE is made of four fields: an MPLS label used to forward the packet to the next router, a Traffic Class field for quality of service, priority, and Explicit Congestion Notification, a bottom of stack flag bit[3], and a time-to-live (LSE-TTL) field having the same purpose as the IP-TTL field, i.e., avoiding forwarding loops.

Labels may be allocated through the *Label Distribution Protocol* (LDP[4]) [23]. 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. A LSR may bind labels to destination prefixes either ($i$) through *ordered LSP control* (default configuration of Juniper routers [24]) or ($ii$), through *independent LSP control* (default configuration of Cisco routers [25, Chap. 4]).

In the former mode, a LSR only binds a label to a prefix if it is local (e.g., a loopback address of the LSR being the exit point of the LSP at the edge of the cloud), or if it has received a label binding proposal from its IGP next-hop towards the given prefix. Juniper routers use this mode as default and only propose labels for loopback IP addresses. In the second mode, the Cisco default one, a LSR creates a label binding for each IGP prefix it has in its RIB (connected, learned, or redistributed within the IGP) and distributes it to all its neighbors. Thus, a label proposal is sent to all neighbors without ensuring that the LSP is enabled up to the exit point of the tunnel. To signal the end of a LSP, the last LSR advertises a terminating label for the corresponding FEC. This label may either be Explicit or Implicit NULL.

### B. MPLS Data Plane and TTL processing

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

- PUSH. The first MPLS router (*Ingress Label Edge Router* – Ingress LER – $PE_1$ on Fig. 1) pushes the LSE in the IP packet, associating so its FEC to a LSP and turning it into an MPLS frame. When pushing the LSE, either the Ingress LER sets the LSE-TTL to an arbitrary value (255, using the `no-ttl-propagate` option, called the *pipe mode*) or it copies the current IP-TTL value into the LSE-TTL (with the `ttl-propagate` option, the default behavior called the *uniform mode*).
- SWAP. 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. While the IP-TTL is not modified, the LSE-TTL is decremented at each hop. If the LSE-TTL expires, the LSR forges an ICMP `time-exceeded` that is sent back to the packet originator. In that case, the LSR may also quote the full MPLS LSE stack of the expired packet in the ICMP `time-exceeded` message [26].
- POP. The *Ending Hop* (EH), the last LSR of the LSP, deletes the LSE, turning the MPLS frame back into an IP packet. Depending on the configuration, two unlabelling modes are possible. The default mode [12] is *Penultimate Hop Popping* (PHP), where the *Penultimate Hop LSR* (PH – $P_3$ in Fig. 1) is in charge of removing the LSE to reduce the load on the Egress. With the *Ultimate Hop Popping* (UHP), the Egress LER ($PE_2$ in Fig. 1) is responsible for the LSE removal, typically to ensure that the Traffic Engineering information (or the VPRN label), if any, is carried up to the LSP end.

When popping the LSE, the EH has to decide the TTL value to copy in the IP header. If the `ttl-propagate` feature has not been disabled, the LSE-TTL will be lower than the IP-TTL and should thus replace the current IP-TTL, while the IP-TTL should be selected otherwise (with the pipe mode). This way, the resulting outgoing TTL cannot be greater than the incoming one. In the latter case, internal hops are not counted, the IP-TTL being unmodified, while they are for the former uniform case.

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 in Cisco PHP configurations [25]), or ($ii$) assuming the Ingress configuration (`ttl-propagate` or not) is the same as the local configuration. This second solution is implemented by JunOS and also in some Cisco UHP configuration. Applying the MIN(IP-TTL, LSE-TTL) allows for maintaining a consistent behavior

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

[3] In order to indicate whether the current LSE is the last in the stack. For the sake of simplicity, we will only consider a single LSE per packet in the remainder of this paper.

[4] Labels might also be distributed with RSVP-TE [21] for traffic engineering purposes. LDP is the most prominent label binding protocol [22], [12] as it is generally the per-default deployment in most MPLS clouds.

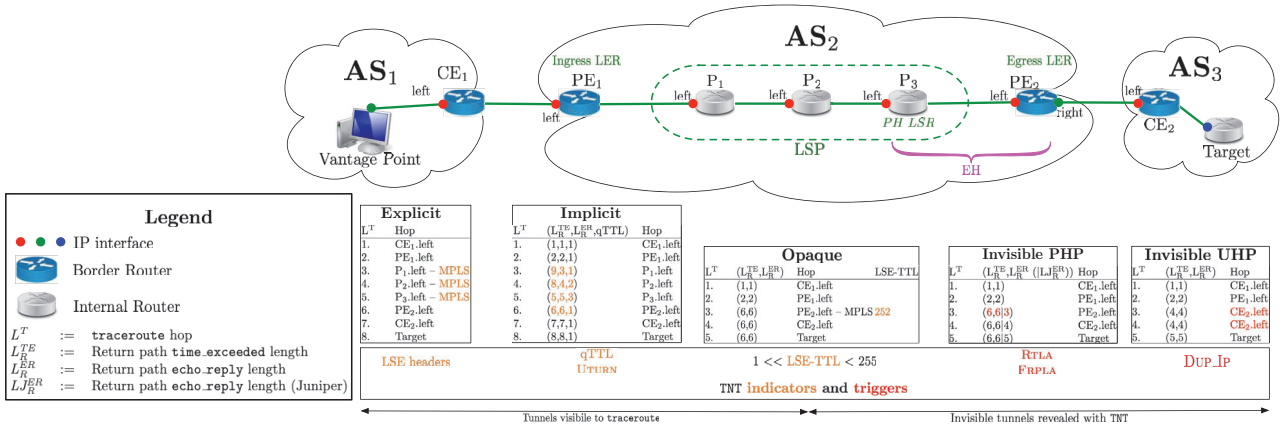


Fig. 1: Illustration of MPLS vocabulary and relationship between MPLS and `traceroute`. The figure is made of three parts. The upper part represents the network topology we use throughout the paper to illustrate concepts. In particular, with respect to MPLS, $P_3$ is the Penultimate Hop (PH). In the case of PHP, $P_3$ is the Ending Hop (EH) and is responsible for removing the LSE. In the case of UHP, the LSE is removed by the Egress LER ($PE_2$). The middle part of the figure presents the MPLS Tunnel taxonomy, as observed with `traceroute` (it is an improvement of the original one proposed by Donnet et al.) 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) is provided. This is used in Sec. V for illustrating `TNT`.

even in the presence of heterogeneous `ttl-propagate` configurations.

ICMP processing in MPLS tunnels varies according to the ICMP type of the message. ICMP *Information messages* (e.g., `echo-reply`) are directly sent to the originator of the `echo-request`. On the contrary, ICMP *Error messages* (e.g., `time-exceeded`) are generally forwarded to the Egress LER that will be in charge of forwarding the packet through its IP plane [11]. Differences between Juniper and Cisco OSes and configurations are discussed in details in an extended version of this paper [20].

## III. Revisiting MPLS Tunnels Taxonomy

According to whether LSRs implement RFC4950 (i.e., ICMP `time-exceeded` quoting MPLS LSE) or not and whether they activate the `ttl-propagate` option or not, MPLS tunnels are more less visible to `traceroute` [11].

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

As illustrated in Fig. 1, Explicit tunnels form the ideal case as all the MPLS information comes natively with `traceroute`. For Implicit tunnels, Donnet et al. [11] have proposed techniques to identify their LSRs based on the way they process ICMP messages and the quotation of the IP-TTL in the `time-exceeded` reply (qTTL and UTURN in Fig. 1).

Opaque tunnels only occur with Cisco routers and are due to LSP ending abruptly, in an improper fashion. As illustrated in Fig. 1, Opaque tunnels and their length can be identified thanks to the LSE-TTL quoted in ICMP `time-exceeded`. Based on large-scale measurements and cross-validation using our GNS3 emulation platform [20], we discovered that the vast majority of Opaque tunnels seems to be caused by Carrier-of-Carriers VPN or similar technologies. Indeed, they provoke an abrupt tunnel ending as the end-to-end bottom label carried to determine the outgoing VPN is not a terminating label.

The `traceroute` behavior for Invisible tunnel differs according to the popping scheme (i.e., PHP or UHP) and the OS, as illustrated in Fig. 1. While Invisible PHP tunnels are identified through path length asymmetry [12] (see Sec. V), Invisible UHP tunnels provoke a duplicated IP (at least with the IOS 15.2). Upon the reception of a packet having an IP-TTL of 1, the Egress LER ($PE_2$ in Fig. 1) does not decrement this TTL, but, rather, 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. This surprising pattern, a duplicated IP at two successive hops, illustrated as "Invisible UHP" in Fig. 1 might be misunderstood as a forwarding loop.

## IV. Hidden Tunnel Revelation

Techniques for revealing the content of Invisible PHP and UHP tunnels are similar. However, in the case of an Invisible PHP tunnel, they can be applied directly as we know both ends of the tunnel (Ingress and Egress LER – see Fig. 1). However, for Invisible UHP the Egress LER is missing from the `traceroute` output (see Fig. 1).

It is, nevertheless, possible with Invisible UHP to infer the outgoing IP interface of the Egress LER (the right interface, in green, on $PE_2$ in Fig. 1). Thanks to its retrieval, `TNT` can force replies from the Egress LER incoming interface (the left one, in red, on $PE_2$ in Fig. 1). This technique, called *buddy*, assumes, for the sake of simplicity although it can be extended, a simple point-to-point connection between the Egress LER and its next-hop. The corresponding IP addresses should then belong to a /31 or a /30 prefix [7], [27] and are called buddies.

With a /30, four IP addresses are available: addresses 0 and 3 are the network and broadcast addresses while addresses 1 and 2 are used for numbering interfaces. Based on that, it is quite straightforward to guess the address of $CE_2$'s buddy (i.e., $PE_2$.right in Fig. 1). If $CE_2$.left corresponds to address 0 (resp. address 3) in a /30, it means that $PE_2$ and $CE_2$ share a /31 and $PE_2$.right is address 1 (resp. address 2) of the /30. However, if $CE_2$.left corresponds to /30 addresses 1, we launch a `ping` towards address 0 within the /30. If an `echo-reply` is received, both interfaces are on a /31 and $PE_2$.right corresponds to address 0. Otherwise, $PE_2$.right belongs to address 2. The same reasoning can be done with address 2 in the /30 for $CE_2$.left.

As ICMP `time-exceeded` typically contains the IP address of the incoming interface having received the expired probe, running a `traceroute` towards the inferred address of $PE_2$.right allows to obtain $PE_2$.left. Once the potential Ingress and Egress LERs are known, we can launch a hidden tunnel revelation technique, i.e., DPR or BRPR [12]. The choice of the technique depends on the way labels have been bound to destination prefixes (see Sec. II-A). It is worth recalling that we can easily discriminate Cisco and Juniper devices using network fingerprinting [28].

On the one hand, with ordered LSP control (Juniper default case), all the external BGP transit traffic goes through MPLS tunnels while the traffic destined to internal prefixes relies on IP forwarding. A single `traceroute` targeting the Egress LER is enough to reveal all LSRs along the LSP. This technique is called *Direct Path Revelation* (DPR). Applying DPR on Fig. 1, `TNT` simply sends probes targeting $PE_2$ revealing $P_1$, $P_2$, and $P_3$ in a row (without labels, as for IP traffic).

On the other hand, with independent LSP control (Cisco default case), LDP is enabled at a wider scope such that each LSR binds labels for each prefix in its IGP RIB. Since `traceroute` naturally reveals the incoming IP interface of each Egress LER, we can apply a recursive `traceroute` approach that targets this last internal prefix to reveal each intermediate hop in a backward fashion from the Egress LER to the Ingress LER. This technique is called *Backward Recursive Path Revelation* (BRPR). Applying BRPR on Fig. 1, we first send a `traceroute` towards $PE_2$ and discover $P_3$. We next send a `traceroute` towards $P_3$ and discover $P_2$ and so on until the Ingress LER is met again.

## V. TNT Design

This section introduces our tool, `TNT` (**T**race the **N**aughty **T**unnels), able to reveal most of MPLS tunnels hidden along a path. `TNT` is built upon Paris Traceroute [17] to mitigate load balancing issues.

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

For each pair of collected IP addresses, `TNT` checks for the presence of tunnels through so-called *indicators* and *triggers*. The former provides reliable indications about the presence of an MPLS tunnel without necessarily requiring additional probing. Generally, indicators suggest uniform tunnels (or to the last hop of an Opaque tunnel), and are basic evidence of visible MPLS presence such as LSEs quoted in the ICMP `time-exceeded` packet (see Sec. V-A for details). Triggers, except DUP_IP, are unsigned values suggesting the presence of Invisible tunnels through a large shifting in path length (see Sec. V-A for more details). When exceeding a given threshold $\mathcal{T}$, triggers fire path revelation methods between the candidate Ingress and Egress LERs as already developed in Sec. IV.

### A. Indicators and Triggers

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

Listing 1: Pseudo-code for checking indicators and triggers

```
if (is_mpls(cur_hop))
  if (T_LSE_TTL < cur_hop.lse_ttl < 255)
    return LSE-TTL #Opaque tunnel
  else
    return LSE #Explicit tunnel

if (cur_hop.qttl > 1)
   return qTTL #Implicit tunnel

if (cur_hop == next_hop)
   return DUP_IP #Invisible UHP tunnel

#inferring path length from raw TTLs
L_R^TE = path_len(cur_hop.ttl_te)
L_R^ER = path_len(cur_hop.ttl_er)
L^T = cur_hop.probe_ttl
diff_te_er = L_R^TE - L_R^ER

if (sign_is_junOS(cur_hop))
   if (diff_te_er ≥ T_RTLA)
      return RTLA #Invisible PHP tunnel
elif (|diff_te_er| > T_URTN)
      return URTN #Implicit tunnel
if (L_R^TE - L^T ≥ T_FRPLA)
   return FRPLA #Invisible PHP tunnel
```

Tunnels indicators are pieces of evidence of MPLS tunnel presence and concern cases where tunnels (or parts of them)

can be directly retrieved from the original `traceroute`. Explicit tunnels are indicated through LSEs directly quoted in the ICMP `time-exceeded` message – See line 5 in Listing 1 and `traceroute` output on Fig. 1. Fig. 1 highlights the main patterns `TNT` looks for firing 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 a quoted LSE-TTL not being equal to an expired value. This abnormal behavior is due to the way labels are handled with Cisco routers, in particular with VPRN tunnel ending. This is illustrated in Fig. 1 where we get a value of 252 because the LSP is actually 3 hops long. This surprising quoted LSE-TTL is an evidence in itself. It is illustrated in lines 2 to 3 in Listing 1, where a hop is tagged as Opaque if the quoted LSE-TTL is between a minimum threshold, $\mathcal{T}_{LSE\_TTL}$(see Sec. VI for fixing a value for the threshold) and 254 (LSE-TTL is initialized to 255).

Implicit tunnels are detected through qTTL and/or UTURN indicators [11]. 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 7 in Listing 1. Second and by default, the UTURN indicator relies on the fact that LSRs send ICMP `time-exceeded` messages to the Egress LER which, in its turn, forwards the packets to the probing source. However, such LSR reply directly to other kinds of probes (e.g., `echo-request`) using their own IP forwarding table, if available. As a result, return paths are generally shorter considering `echo-reply` messages than regarding `time-exceeded` replies. Thereby, the UTURN indicator reflects this difference in these lengths.

On the one hand, such indicators are generally pieces of evidence of visible MPLS tunnels not requiring further probing (except for some LSE values, shown at line 2, also being triggers for Opaque tunnels). On the other hand, triggers are patterns suggesting the presence of Invisible tunnels (both PHP and UHP) that could be revealed using additional probing (see Sec. IV). In this category, `TNT` looks first for potential Invisible UHP tunnels (line 10). As explained in Sec. III, they occur with Cisco routers using IOS 15.2 and result in a duplicate IP address in the trace output ($CE_2$ in Fig. 1).

The two remaining triggers, RTLA (Return Tunnel Length Analysis) and FRPLA (Forward/Return Path Analysis) [12]), rely on path lengths. More precisely, RTLA is the difference between the `time-exceeded` and the `echo-reply` return path lengths, while FRPLA is the difference between the forward and the return path lengths of `traceroute` probes and associated replies. Both triggers are based on the idea that replies sent back to the vantage point are also likely to cross back the MPLS cloud, which will apply the MIN(IP-TTL, LSE-TTL) operation at the EH of the return tunnel. In the absence of Invisible tunnel, we expect those triggers to have a value equal or close to 0. Therefore, any significant deviation from this value is interpreted as the potential presence of an Invisible MPLS cloud, and thus, fires additional path revelation techniques (see Sec. IV).

To check for those triggers, we first extract the key distances thanks to the IP-TTLs in replies received by the vantage point (lines 14 to 16 in Listing 1). Since RTLA only works with JunOS routers [12], prior to estimating the triggers, `TNT` uses network fingerprinting [28] to determine the router brand of the potential Egress LER (line 19 in Listing 1).

In the presence of a JunOS hardware (line 19), `time-exceeded` and `echo-reply` packets have different initial TTL values [28], and the RTLA trigger can exploit the TTL gap between those two kinds of messages caused by the MIN(IP-TTL, LSE-TTL) behavior at the Egress LER. Indeed, the $L_R^{ER}$ is longer than the $LJ_R^{TE}$ as the MIN operation considers a differentiated pick. This difference represents the number of LSRs in the return LSP, and is compared to a predefined threshold $\mathcal{T}_{\text{RTLA}}$(line 20). This threshold (see Sec. VI for the parameter calibration) filters very short LSPs. Finally, if the signature does not correspond to JunOS, `TNT` fallback to the UTURN indicator (see line 23).

FRPLA is more generic and applies thus to any configuration. FRPLA allows to compare, at the AS granularity, the forward (i.e., $L^T$) and return paths (i.e., $L_R^{TE}$) length distribution. 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 24). This is illustrated in Fig. 1 ("Invisible PHP") in which $L^T$ is 3 while $L_R^{TE}$ is equal to 6, leading so to an estimation of the return tunnel length of 3. In general, when no IP hop 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 handle path asymmetry, `TNT` uses a threshold, $\mathcal{T}_{\text{FRPLA}} > 0$, to avoid generating numerous false positives.

### B. *TNT Limits*

By using GNS3, we aimed first at verifying that the inference assumptions considered in the wild are correct and reproducible under 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, focusing on the POP operation) of some common OSes used by many real routers. Details of experiments done with GNS3 are provided in the extended version of this paper [20].

Table I provides a summary of `TNT` capacities considering several MPLS usages in standard configurations. For example, it shows that `TNT` is able to discriminate between Cisco Invisible UHP and PHP tunnels while it is not the case for Juniper

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

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

routers. Indeed, for both UHP/PHP Juniper configurations, the trigger and the revelation methods are the same (RTLA and DPR). Moreover, we also show for which cases our basic set of techniques need to be extended for enabling revelation and distinction among different classes. We use the symbol ++ to enforce these new requirements that are described in detail in our technical report. Revealing UHP Cisco tunnels requires in particular to extend BRPR with the additional `buddy()` function and UDP probing. Another example is LSE-TTL++ that simply means that the quoted LSE-TTL is equal to 255.

As shown by Table I, VPRN tunnels may be classified by TNT. In practice, they conduct to distinct surprising patterns: opaque tunnels for Cisco routers and non-monotonic TTL evolution with Juniper routers.

Our GNS3 platform shows that VPRN content cannot be revealed with TNT, while other Opaque tunnels configurations (i.e., routing devices heterogeneity, BGP edge configuration) can. The absence of content revelation can be explained by the IP address collected by TNT from the source IP field in the ICMP reply. Usually, the collected address is the one of the incoming interface of the Egress PE, while in the VPRN case, it is the one assigned to the interface linked to the VRF. In practice, this corresponds to the outgoing interface towards the VPN at the customer's side. Said otherwise, TNT collects the outgoing address instead of the incoming one. While the outgoing address usually allows TNT to get the incoming one, it turns out to be impossible within a VPRN (details are provided in [20] – in particular, we explain why such opaque tunnels are not revealed but are detectable and can be distinguished from other kinds of P2P tunnels).

## VI. TNT CALIBRATION

For calibrating various TNT thresholds, we deployed TNT on three vantage points (VPs) over the Archipelago infrastructure [19]. VPs were located in Europe (Belgium), North America (San Diego), and Asia (Tokyo).

TNT was run on April 6th, 2018 towards a set of 10,000 destinations (randomly chosen among the whole set of Archipelago destinations list). Each VP had its own list of destinations, without any overlapping.

From indicators and triggers described in Sec. V-A, it is obvious that UTURN is equivalent to RTLA for Juniper routers. However, the $\mathcal{T}_{\text{UTURN}}$will not have the same value than $\mathcal{T}_{\text{RTLA}}$. $\mathcal{T}_{\text{UTURN}}= 0$ by design as any difference between `echo-reply` and `time-exceeded` replies for the Cisco router signature indicates LSE-/IP-TTL shifting. In practice, we reinforce the condition by looking for at least two consecutive hops having a cumulated UTURN $\geq 3$. Regarding LSE-TTL quoted in `time-exceeded`, we have observed [20] that it oscillates between 236 and 254. A value of 236 for $\mathcal{T}_{LSE_TTL}$is then enough to detect the presence of an Opaque tunnel.

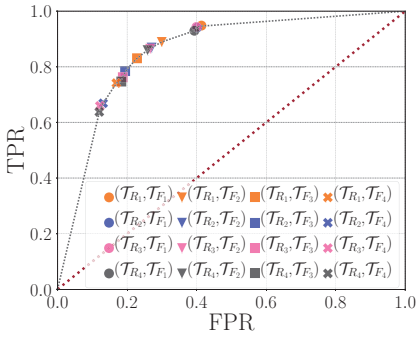


Fig. 2: Receiver operating characteristic (ROC) curve of TNT trigger thresholds (FRPLA and RTLA). $\mathcal{T}_{R_x}$ refers to $\mathcal{T}_{\text{RTLA}}= x$, while $\mathcal{T}_{F_y}$ to $\mathcal{T}_{\text{FRPLA}}= y$.

For our tests, we varied $\mathcal{T}_{\text{RTLA}}$and $\mathcal{T}_{\text{FRPLA}}$between 1 and 4. A full measurement campaign was launched for each pair of parameter value (thus, a total of 16 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/cost tradeoff of the two threshold values.

With the help of well calibrated thresholds, the results associated to FRPLA and RTLA triggers allow for a binary classification. These triggers provide a prediction, while the results of additional probing give the true facts when some conditions apply, 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. 2. 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). In this analysis the brute force data provides the ground data that we consider reliable enough; revelation is fired at each hop and, if nothing is revealed, we consider that there is no tunnel (we do not consider inconclusive cases as TNT did not cross the potential Ingress or Egress LER). 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 extrapolates ground measurement results.

We observe that the results are essentially positive for TNT and its two firing thresholds, $\mathcal{T}_{\text{RTLA}}$and $\mathcal{T}_{\text{FRPLA}}$. The best calibration combination, between ($\mathcal{T}_{R_1}$, $\mathcal{T}_{F_3}$) and ($\mathcal{T}_{R_2}$, $\mathcal{T}_{F_3}$), provides results offering a compromise close to 80%-20%: while we expect to reveal at least 80% of existing tunnels (MPLS artificial direct links), TNT has a controlled overhead of 20%, i.e., it fires useless additional probing for an average limited to two actual IP links over ten.

We also notice that the amount of probes required for actually revealing the content of Invisible tunnels remains almost stable, whatever the values for $\mathcal{T}_{\text{FRPLA}}$and $\mathcal{T}_{\text{RTLA}}$. The additional traffic generated by erroneous triggers or by inconclusive revelation decreases while $\mathcal{T}_{\text{FRPLA}}$increases. The overhead of TNT is quite limited compared to a standard active campaign (i.e., considering the overall information gathered).

When calibrated properly in order to limit both useless probing and missed tunnels (e.g., $\mathcal{T}_{R_1}$, $\mathcal{T}_{F_3}$), TNT can reveal 80% of hidden MPLS tunnels by generating less than 10% of additional probing. Next section will show that these Invisible tunnels account for more than 15% of MPLS tunnels in general (absolute values are given in Table II).

## VII. Tunnels Quantification With TNT

We deployed TNT on the Archipelago infrastructure [19] on April 23rd, 2018 with parameters $\mathcal{T}_{\text{FRPLA}}$fixed to 3 and $\mathcal{T}_{\text{RTLA}}$to 1, according to results discussed in Sec. VI.

TNT has been deployed over 28 vantage points, scattered all around the world: Europe (9), North America (11), South America (1), Asia (4), and Australia (3). The overall set of destinations, nearly 2,800,000 IP addresses, is inherited from the Archipelago dataset and spread over the 28 VPs to speed up the probing process.

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 [28]. Our dataset and our post-processing scripts are freely available.[2]

Table II provides the number of MPLS tunnels discovered by TNT, per tunnel class as indicated in the first column. The indicators/triggers are provided, as well as the additional revelation technique used. Without any surprise, Explicit tunnels are the most prevalent class (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 is subject to false positive (implicit UTURN tunnels are likely to be overestimated because of possible confusion with RTLA for Juniper routers), while qTTL is much more reliable [29]. Compared to previous works, it is clear that this class is not as prevalent as expected at the time, both because we corrected and improved our methodology, and also because the RFC4950 is likely to be more and more deployed.

Opaque tunnels are less prevalent (1.7% of tunnels discovered). It is worth noticing that additional revelation techniques (DPR or BRPR) does not perform well with such tunnels. The content of 98% of Opaque tunnels cannot be revealed, suggesting so that the vast majority of Opaque tunnels seems to arise due to Cisco VPRNs.

The proportion of Invisible tunnels is not negligible (16% of tunnels in our dataset). These measurements clearly contradict our previous work suggesting that Invisible tunnels were probably 40 to 50 times less numerous than Explicit ones [11, Sec. 8]. More precisely, Invisible PHP is the most prominent configuration (87% of Invisible tunnels belongs to the Invisible PHP class), confirming so our last survey [12]. RTLA appears as being the most efficient trigger. This is partially due to the order of triggers in the TNT code as it favors a high ranked trigger (RTLA) compared to low ranked one (FRPLA– in case both apply, we prefer to use the most reliable, i.e., the less subject to any interference such as BGP asymmetry). DPR works better than BRPR, which is obvious as it is triggered by RTLA (Juniper routers). For Invisible UHP, less numerous, it is worth noticing that the buddy, prior to BRPR or DPR revelation, was required in only 25% of the cases. In other cases, a simple BRPR or DPR revelation was enough to get the tunnel content.

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, ISP may deploy both Juniper and Cisco hardware without any homogeneous prefixes distribution (i.e., local prefixes for Juniper, all prefixes for Cisco – See Sec. II-A for details). Note that it is also possible that the UHP and PHP label popping techniques co-exist when using our backward recursive path revelation (BRPR). Although not explained in Sec. V for clarity reasons, TNT can deal with such complex situations, making the tool robust to pitfalls encountered in the wild (5% of the Invisible tunnels encountered). The column labeled "1HOP" corresponds to single LSR tunnels where DPR and BRPR cannot be distinguished.

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

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

Finally, Fig. 3 provides the distribution of path length with standard `traceroute` and with TNT. We clearly see that TNT leads to a shift of the distribution towards the right (longer

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

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

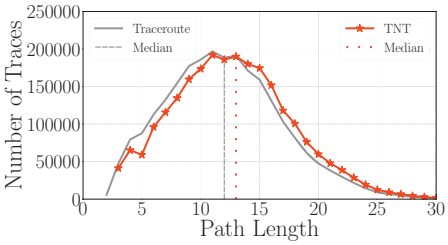


Fig. 3: Path length distribution correction with TNT.

paths). This shift is lower than the median length of tunnels given in the last column of Table II because all traces are taken into account, even the ones with no tunnels. Vanaubel et al. [12] have shown how revealing hidden tunnels also impact standard Internet model metrics.

## VIII. Conclusion

In this paper, we first revised the MPLS classification proposed by Donnet et al. [11]. Second, we introduced TNT (**T**race the **N**aughty **T**unnels), an extension to Paris traceroute for revealing most MPLS tunnels along a path. Our fully integrated tool reveals, or at least detect, all kinds of tunnels in two simple stages. TNT relies on indicators and triggers to classify and possibly tag tunnels as hidden, and then launches additional probing to reveal their content. TNT provides the ability to unveil the MPLS ecosystem deployed by ISP. Recent works have indeed shown that MPLS is largely deployed by most ISP [11], [22], [13]. By running TNT periodically from largely distributed measurement platforms (e.g., Archipelago, RIPE Atlas), we expect to see numerous researches using our tool to correct graph properties and models used to better understand the actual Internet topology.

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

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

[4] 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.

[5] K. Keys, "Internet-scale IP alias resolution techniques," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 50–55, January 2010.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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

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

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

[17] 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.

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

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

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

[21] 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.

[22] 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.

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

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

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

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

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

[28] 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.

[29] G. Davila Revelo, M. A. Ricci, B. Donnet, and J. I. Alvarez-Hamelin, "Unveiling the MPLS structure on Internet topology," in *Proc. Traffic Monitoring and Analysis Workshop (TMA)*, April 2016.