Network Fingerprinting: TTL-Based Router Signatures

Yves Vanaubel
Université de Liège
Belgium
yves.vanaubel@ulg.ac.be

Jean-Jacques Pansiot
Université de Strasbourg
France
pansiot@unistra.fr

Pascal Mérindol
Université de Strasbourg
France
merindol@unistra.fr

Benoit Donnet
Université de Liège
Belgium
benoit.donnet@ulg.ac.be

ABSTRACT

Fingerprinting networking equipment has many potential applications and benefits in network management and security. More generally, it is useful for the understanding of network structures and their behaviors. In this paper, we describe a simple fingerprinting mechanism based on the initial TTL values used by routers to reply to various probing messages. We show that main classes obtained using this simple mechanism are meaningful to distinguish routers platforms. Besides, it comes at a very low additional cost compared to standard active topology discovery measurements. As a proof of concept, we apply our method to gain more insight on the behavior of MPLS routers and to, thus, more accurately quantify their visible/invisible deployment.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology

General Terms

Measurement

Keywords

network discovery, fingerprinting, MPLS, router signatures, initial TTL

1. INTRODUCTION

Fingerprinting [1, 2] refers to the act of dividing network equipment into disjoint classes by analyzing messages sent by that equipment, usually in response to some form of active probing. Those classes may correspond, for instance, to router operating system (OS), router brand, or router configuration. Providing such a fingerprinting is useful for several applications and studies. Indeed, for instance in network management, if the fingerprinting is based on router OS, it may help in listing the network nodes and identifying vulnerable hosts in terms of security and fault tolerance [1, 3]. It may also help in identifying which nodes have an abnormal behavior (e.g., delay, packets drop/modification, etc). In network topology discovery [4], fingerprinting could find a suitable usage to understand how various types of equipment are interconnected. Indeed, obtaining the router level map of the topology from `traceroute` data requires an additional probing intensive step: *alias resolution* [5]. Router fingerprinting may drastically speed up this step, since IP addresses belonging to different classes cannot be aliases and, so, do not require to be further probed for alias resolution. Another interesting application is to understand whether IP networks are heterogenous in terms of hardware and software at different scales (e.g., temporal to study the evolution and structural to understand internal structure of autonomous systems). Indeed, an accurate fingerprinting technique may allow one to distinguish router OS among a given brand.

However, fingerprinting can be costly and possibly intrusive as it could require many probes [1]. In this case, fingerprinting could be a time consuming process using undue network resources. Moreover, too many probes towards a network node or a subnet could be seen as remote host scanning and, consequently, be filtered.

In this paper, we present a fingerprinting method that is a companion to `traceroute`-like exploration. Our method is simple[1], requires few additional probes to `traceroute` ones, but still allows for classifying Internet routers based on their hardware and OS. Our fingerprinting method infers initial TTL values [6, 7] used by routers when building their different kinds of reply packets. We call this set of TTL values *router signatures*. Router signatures are meaningful for fingerprinting as the initial TTL values vary not only between different router platforms but also depending on the protocol and the type of message (error versus standard replies for instance). Indeed, no specific default value has been standardized for the TTL field.[2]

We consider a router signature as a n-tuple made of n initial TTL values. Those n TTL values are derived from TTL included in different types of probe replies. The number and

IMC'13, October 23–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2504730.2504761.

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

[2]It is worth to notice that RFC1700 recommends to use 64 as initial TTL value [8]. This is however not followed by most router manufacturers.

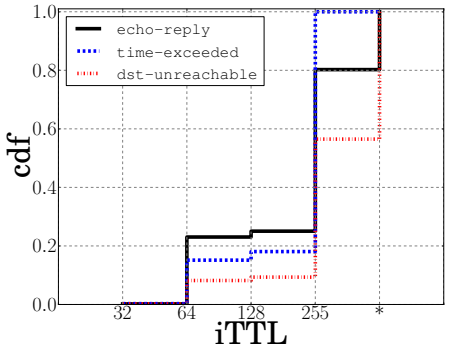


Figure 1: Initial TTL distribution (∗ refers to non-responding routers)

the variety of probes sent give the actual value of n. In this preliminary work, we focus on $n = 2$ TTL values, but we already envision longer (i.e., $n > 2$) signatures to provide better distributions among router's OS. Indeed, the more discriminating, the more meaningful the significant classes. Note that it is not that easy since the OS market is known to be not uniformly shared. Thus, for an application such as providing a pre-partition to speed up alias resolution process, the interest may be limited at the granularity of the partition.

Based on a large-scale measurement campaign, we first demonstrate that our router signatures are consistent among measurement points. After providing some general distribution results, we illustrate how it can be used in the context of topology discovery [4] or active probing in general. In particular, it can be useful to determine if measurements are biased due to router type dependency. In this paper, we focus on MPLS tunnels identification and validation [9, 10]. We try to map the behavior of several MPLS tunnel classes [10] to our set of router signatures. We are thus able to improve our previous study [10] on MPLS quantification thanks to TTL-based fingerprinting. We believe that any active probing tool can take benefit from our simple fingerprinting proposal.

The remainder of this paper is organized as follows: Sec. 2 discusses our TTL-based signatures and our methodology; Sec. 3 shows how those classes can be helpful to improve our previous MPLS tunnels classification and quantification; Sec. 4 positions this paper regarding the state of the art; finally, Sec. 5 concludes this paper by providing discussions on ongoing works and perspectives for further works.

2. ROUTER SIGNATURES

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

The IP packet header contains a *time-to-live* (TTL) field used to avoid packets looping forever when a routing loop occurs. This 8-bit field is set by the originating host/router to an *initial value* that is usually and nearly always a power of 2 in the list 32 (or 30), 64, 128, and 255. The TTL field is decremented by one at each intermediate node along the path the packet takes. When the TTL value is one, the router determines that the packet has consumed sufficient resources in the network, drops it, and informs the packet source by sending back an *Internet Control Message Protocol* (ICMP) `time-exceeded` message. `traceroute` [11] is based on this simple behavior.

In this paper, we need to determine the initial TTL ($iTTL$) of a received packet. Obviously, the (known) received value is equal to $iTTL - \#hops$ where $\#hops$ is the number of hops between the sender and the receiver. It is worth to notice that $\#hops < 30$ most of the time (99.8% of the paths, in our dataset, are less than 30 hops). Therefore, the iTTL value can be estimated as the smallest number in $32, 64, 128, 255$ that is larger than the received value. In very infrequent cases, an iTTL of 64 together with a very long route ($\#hops > 32$) would give an incorrect guess of 32 instead of 64, e.g., a route of length 34 would be interpreted as a route of length 2. Those cases could be managed during a `traceroute` campaign by looking at the number of hops of the forward route. A difference between the number of forward and backward hops close to 32 would indicate that the iTTL is 64 instead of 32.

A router *signature* is made of a n-tuple of n iTTLs, those iTTLs being retrieved from different ICMP messages. Our basic pair-signature (with $n = 2$) simply uses the iTTL of two different messages: a `time-exceeded` message elicited by a `traceroute` probe, and an `echo-reply` message obtained from an `echo-request` probe. Quite surprisingly, for the same ICMP protocol, a significant proportion of nodes use two different iTTL values for these two messages, as shown on Fig. 1.[3] We also tried to add a third iTTL to our signatures: a `destination-unreachable` message elicited by a UDP probe. As shown on Fig. 1, more than 40% of routers do not respond to such probes. Mostly, our basic signatures are thus extended with an absence of response. We decided to not take the `destination-unreachable` iTTL into account in this preliminary study. However, the information it brings extends the number of possible signatures and we think it could be helpful for alias resolution in particular. In future works, we envision to consider more ICMP reply types and also different IP header fields. We already observed that the iTTL does not only depend on the reply type (error or standard replies) nor the answer origin (the central or the per interface processors).

In theory, using n probes, we may have up to $4 \times 5^{n-1}$ different signatures since we can count the absence of answer for `echo-reply` (i.e., a $*$) as a valuable pattern. It is worth to notice that the "4" before the multiplication sign is due to the fact that ICMP `time-exceeded` messages are our basic probing mechanism, i.e., `time-exceeded` messages are used to direct subsequent probes (`echo-request`, UDP packets, ...). This means that the $*$ does not count for `time-exceeded` messages and, only $|\{255,128,64,32\}|$ patterns are available for such replies.

2.1 Measurement campaign

We used Paris Traceroute [12] with ICMP `echo-request` packets to collect IP level paths. Each ICMP `time-exceeded` packet received is used to build the first component of a router signature. In addition, for each IP interface discovered, we sent 6 ICMP `echo-request` probes (in particular to ensure the robust meaning of a $*$ and help our MPLS discovery). We used the ICMP `echo-reply` packets received to complete the second component of our router signature.

[3]The methodology of our measurement campaign will be given in Sec. 2.1.

Note that, each IP address collected is pinged six times only once per vantage point (i.e., when it is discovered for the first time).

We perform our measurement study with a team of 200 randomly selected PlanetLab vantage points (VPs). Of the 200 VPs, 121 were located within the US; 10 VPs were located in Europe, and the other 69 in different countries. We randomly selected 1,000,000 destinations in the Archipelago [13] target list and evenly divided this target list among our VPs team. The dataset has been collected between January 8^{th}, 2013 and January 10^{th}, 2013 using scamper [14]. Once the data has been collected, we consider each IP address only once and associate this unique address to its signature. We so gathered 335,646 distinct IP addresses.

2.2 Measurement Cost

The additional overhead of a measurement campaign to fingerprint a set of routers found by a `traceroute` measurement campaign comes at a low cost: each discovered IP address must be probed with k ICMP `echo-request` messages, where k is a robustness parameter. Usually, many traces discover the same IP addresses that need to be probed only once [15]. So, at worst, this fingerprinting needs k more messages per IP address than `traceroute` only, and, on the average, much less. In our measurement campaign (where $k = 6$), 13.437.896 `traceroute` responses and 14.803.614 `ping` responses have been received. That is with our fingerprinting method, and on the average, a probed node sends about the same number of `time-exceeded` messages and `echo-reply` messages. This number could be further reduced by using a smaller robustness factor or by adding some extra cooperation between VPs in order to avoid as much as possible pinging the same IP address several times.

2.3 Signatures Consistency

The objective behind fingerprinting is to obtain a signature that depends only on the probed node. To verify that fact, we compared the signatures of IP addresses observed by at least two distinct VPs (that is through distinct `traceroute` and `ping` probes). Note that we consider only IP addresses that responded to `traceroute` probes, but some of them do not always respond to `ping` probes (i.e., the second component of the signature may take the value $*$). For the IP addresses seen from several VPs, we classify our signatures into three categories:

- *coherent*: the same signature is observed for a given router interface among measurements done by all VPs. This is the perfect case. Coherent signatures are observed in 95.92% of the cases.
- *weakly incoherent*: from some VPs, the signature is of the type $< x, y >$ while it is $< x, * >$ for some others. This concerns 3.94% of our signatures.
- *incoherent*: several different signatures are observed for a given router interface among measurements done by all VPs. This is the worst case but it is also very infrequent (0.14% of the cases).

Weakly incoherent signatures can be explained by two phenomena: some nodes may not respond to `ping` at some time, for example because of overloading, rate limiting, or filtering inducing the `echo-reply` lost on some paths [16]. For the 3.94% of weakly incoherent signatures we observed

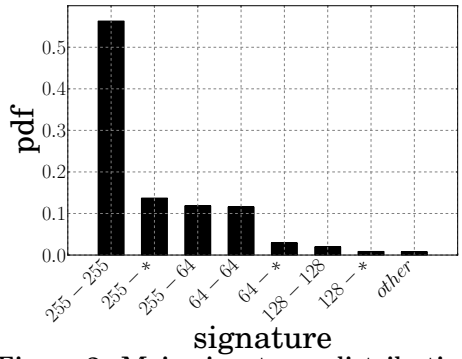


Figure 2: Main signatures distribution

in our dataset, we decide to keep only the best signature (i.e., the complete one). We therefore replace each weakly incoherent signatures by the complete one. After this operation, around 17.5% of the entire set of IP addresses collected in our campaign were still associated to incomplete signatures (of the type $< x, * >$). Most of them (i.e., 10.19%) are seen by several VPs, while the remaining IP addresses (i.e., 7.31%) are seen by a single VP. This latter value could be further reduced by trying to `ping` those IP addresses from other VPs.

Incoherent signatures may be explained by some artifacts (due to their extremely low proportion). Some middleboxes may rewrite the TTL field [17, 18, 19], or the same IP address may correspond to different nodes depending on the network location (*anycast* address). There is also the previously mentioned possible ambiguity between iTTL values 32 and 64.

2.4 Signatures Distribution

While many different platforms could correspond to the same signature, we know the signature of some well known platforms (to this purpose, we performed a bunch of tests in an emulation lab). For instance, Cisco routers generate signature $< 255, 255 >$ while, for Juniper routers, we have $< 255, 64 >$ with Junos and $< 128, 128 >$ with JunosE. Some Brocade and Alcatel equipment together with some Linux boxes result in a $< 64, 64 >$ signature. Although these signatures encompass the main router platforms, it would be very interesting to have a more complete correspondence between platforms and signatures. Obviously, when restricted to our 2-tuple, several very different platforms may have the same signature. So a more accurate signature, i.e., an n-tuple with $n > 2$, would be helpful. We already did some preliminary work analyzing signatures extended with other types of message (`destination-unreachable` in particular) or some other criterion, such as the ICMP messages size. However, since most routers come from a few major vendors, we cannot expect to partition the network nodes into a very fine-grained classification using this type of fingerprinting. This is why, in this study, we only consider this basic 2-tuple as a proof of concept that may be generalized by further study.

Fig. 2 illustrates the distribution of the main router signatures. The first class $< 255, 255 >$, that includes Cisco routers is largely dominant, corresponding to more than 50% of nodes. The fourth class $< 64, 64 >$, that includes several vendors or OSes (including Linux), and the third class $< 255, 64 >$ that includes Juniper routers running JunOS

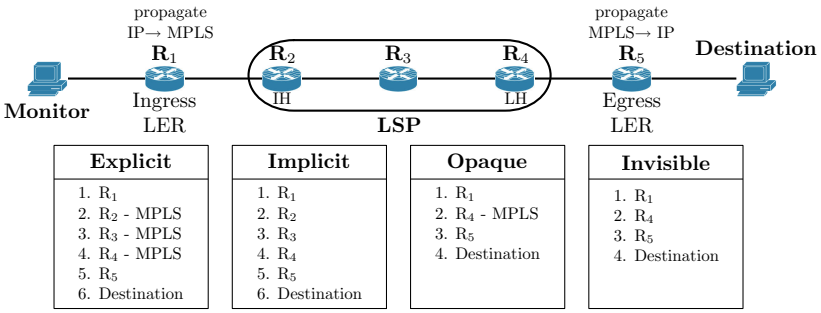


Figure 3: Example of MPLS tunnel

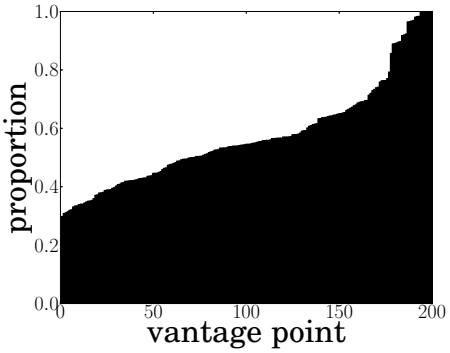


Figure 4: Proportion of paths, per VP, having at least one MPLS tunnel

have about 11% each. The second most frequent class $< 255, * >$, with about 15%, corresponds to an incomplete signature and is probably mostly made of nodes belonging, actually, to $< 255, 255 >$ or $< 255, 64 >$ but did not respond to `ping` for various reasons [16]. The class $< 128, 128 >$, including Juniper platforms running the JunosE system, is around 3% while the remaining classes are either incomplete or very rare. Therefore, at a global scale, our fingerprinting technique seems to reflect the market distribution.

To summarize, we first observe, that among different brands and OSes, routing devices use distinct iTTL values, and, second, we also notice that a single device can use multiple iTTL values (at least, this is the case for Juniper routers). We now focus on a specific use case illustrating the technical interest of a classification based on such observations.

3. MPLS USE CASE

Multiprotocol Label Switching (MPLS) [20] is increasingly deployed by ISPs to provide attractive services such as virtual private networks and traffic engineering. It is therefore interesting to have some insights on MPLS technologies in the Internet. However, MPLS tunnels may hide IP-level information by masking MPLS routers from `traceroute`. We propose to use our fingerprinting method to refine information on our previous tunnels deployment analysis [10]. We will show, using MPLS as an example, that our method could be used to determine whether a feature (here MPLS characteristics) is independent of the router type. Moreover, our fingerprinting method may also be extended to check if a given sample of routers is representative of the Internet heterogeneity.

3.1 MPLS Tunnels Signatures

The MPLS architecture is based on labels: an IP router inserts one or more 32-bit *label stack entries* (LSE – that contains a label, a TTL field called LSE-TTL, and a type-of-service field) into a packet, before the IP header, that determines the forwarding actions made by subsequent MPLS *Label Switching Routers* (LSRs) in the network. A series of LSRs connected together form a *Label Switched Path* (LSP).

In an MPLS network, packets are forwarded using an exact match lookup of the 20-bit label 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.

Fig. 3 illustrates the general behavior of an MPLS tunnel. Router R_1 is the entry of the MPLS tunnel and is the first router to *push* an MPLS label; we call this router the *ingress Label Edge Router* (LER). Router R_2 is the first LSR where the incoming packet includes a LSE; we call this router the *ingress hop* (IH). Router R_4 is the last router that *pops* the MPLS label; we call this router the *last hop* (LH). At least for Cisco routers, most of the time the LH router is located one hop before the *egress LER* due to the use of *penultimate hop popping* (PHP) [20].

Similarly to the IP-TTL, the LSE-TTL field is decremented by LSR that may send ICMP `time-exceeded` messages when the LSE-TTL expires. In order to debug networks where MPLS is deployed, routers may also implement RFC4950 [21], an extension to ICMP that specify that a LSR should embed the MPLS label stack of the incoming packet into an ICMP `time-exceeded` message. The first MPLS router of an LSP may copy the IP-TTL value to the LSE-TTL field rather than setting the LSE-TTL to an arbitrary value such as 255. That is the ingress LER uses TTL propagation. During a `traceroute`, LSRs along the LSP will reveal themselves via ICMP messages even if they do not implement RFC4950. Operators configure this action using the `ttl-propagate` option provided by the router OS.

Based on those two MPLS transparency features, we previously proposed an MPLS taxonomy made of two-by-two classes [10]. Fig. 3 illustrates those classes that are: *explicit tunnels* (i.e., `ttl-propagate` and RFC4950 are enabled), *implicit tunnels* (i.e., the router that pushes the MPLS label enables the `ttl-propagate` option but LSRs do not implement RFC4950), *opaque tunnels* (i.e., the LH implements RFC4950 but the ingress LER does not enable the `ttl-propagate` option), and, finally, *invisible tunnels* (i.e., the ingress LER does not enable the `ttl-propagate` option and RFC4950 is not implemented by the LH router). Our previous work provides MPLS signatures detection for revealing implicit and opaque tunnels based on three main patterns:

1. the quoted IP-TTL (*qTTL*) in ICMP `time-exceeded` messages[4]. A qTTL > 1 will likely reveal the `ttl-propagate` option at the ingress LER of an LSP. For each subsequent `traceroute` probe within an LSP, the

[4]These kind of replies should contain the quotation of the original IP header triggering the error message. Look at RFC 792 and 1812.

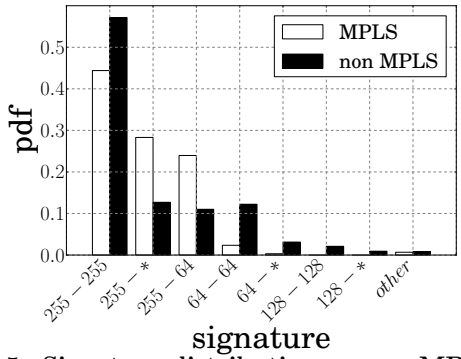


Figure 5: Signature distribution among MPLS and non MPLS routers

qTTL will be one greater resulting in an increasing sequence of qTTL values in `traceroute`;

2. #hops differences with the IP-TTL in `echo-reply` messages (*uturn*). It relies on the fact that LSRs along an LSP present an *original label stack* default routing behavior: when the LSE-TTL expires, an LSR first sends the `time-exceeded` reply to the Egress LER which then forwards the reply on its own to the probing source [5], while an LSR replies to other probes using its own IP routing table if available.

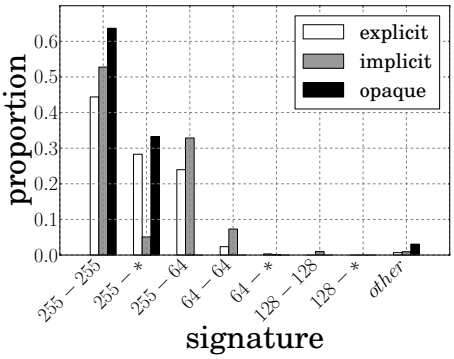
3. opaque tunnels are revealed through an *abnormal* LSE-TTL (1 <LSE-TTL< 255) returned by the LH in the `time-exceeded` reply.

Generally speaking, Fig. 4 shows that a large proportion of paths hits one or more MPLS tunnels: from about half of the VPs, at least half of the paths reveal MPLS tunnels. Explicit and implicit tunnels are by far the most frequent (respectively 46,657 and 61,054 tunnels corresponding to roughly 40% and 60% of IP addresses belonging to tunnels and, altogether, 17% of IP addresses collected). However, note that most implicit tunnels are discovered using a probing inference heuristic (uturn). This could lead to false positive or false negative tunnels. Besides, opaque tunnels are rather rare (and so subject to weak statistics – 523 opaque tunnels), and, by definition, we do not have any hints to find the invisible ones. In our previous MPLS work, we tried to extrapolate the invisible tunnels quantity using uniform linear rules over opaque and implicit ones. We made assumptions regarding RFC4950 and `ttl-propagate` independence, and did not look at any correlation between those features and the router OS. We intend here to determine how we can extend and possibly validate this work using our TTL-signatures, particularly by checking the existence of a correlation between our MPLS patterns and the router OS.

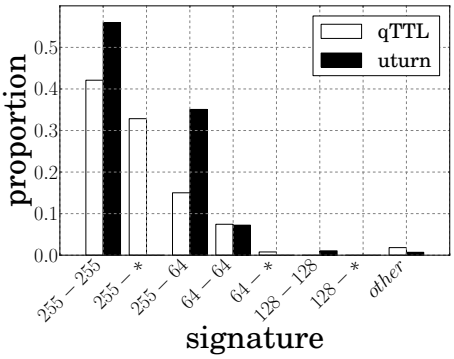
3.2 MPLS Tunnels TTL-Classification

Fig. 5 highlights the difference in frequency of our signatures between MPLS IP addresses (i.e., tagged as such by `traceroute` – it does not include implicit tunnels) and non MPLS IP addresses. The first striking difference is that the signature $< 64, 64 >$ is much less prevalent in MPLS visible networks. This could be explained by the fact that

[5]Look at the `mpls ip ttl-expiration pop` command of Cisco routers



(a) Main tunnel classes



(b) Implicit sub-classes

Figure 6: Router Signature distribution among MPLS tunnel classes

this signature corresponds to a variety of middleboxes and probably less to high-end routers commonly used in MPLS networks. The second lesson is that the dominant share of $< 255, 255 >$ in non MPLS networks is less dominant for MPLS, while signatures $< 255, 64 >$ and $< 255, * >$ increase their share. The increase of $< 255, * >$ may be due to LSRs that do not have a complete IP routing table and thus cannot reply with an `echo-reply` message. Recall that, in this case, an error message such as a `time-exceeded` message is usually propagated to the end of the tunnel before being forwarded to its destination (this is the behavior captured by our uturn heuristic). The increase of $< 255, 64 >$ signatures is likely to balance the decrease of $< 255, 255 >$: it seems that $< 255, 64 >$ routers (e.g., Juniper ones) increase their market position for MPLS operations (compared to the previous ratio $< 255, 255 > / < 255, 64 >$ at a global scale).

Each of our tunnel classes exhibits a specific router signature distribution. Fig. 6(a) presents those distributions. The X-axis gives the various signatures, while the Y-axis shows the proportion of tunnels, in a given MPLS class, that exhibits a given signature. From Fig. 6(a), we see that opaque tunnels are only characterized by signatures $< 255, 255 >$ and $< 255, * >$. This property shows a bias in the linear extrapolation we used to quantify invisible tunnels [10]. We assumed that the LH router would always insert a label stack into the ICMP `time-exceeded` message if it implements RFC4950. Consequently, the difference between an opaque and an invisible tunnel was based on RFC4950 implementation on the LH router. Our new results clearly demonstrate that this is not the case (see Fig. 6(a)): if the LH does not belong to $< 255, 255 >$ class (or its incomplete

companion $< 255, * >$), it will not insert a label stack in the ICMP message even if it implements RFC4950. Hence, the tunnel will actually be invisible. We verified this on a virtual testbed: considering an LSP made of Juniper OS compliant with RFC 4950 (but where the ingress LER does not propagate the IP-TTL), one may not discover any visible tunnels (there are only invisible tunnels, no opaque ones). Moreover, with Cisco OS, the same LSP will appear as opaque or invisible depending on the PHP behavior associated to the probed IP address. It seems that the opaque tunnels are more the exception and invisible tunnels the rule. This means that the ratio invisible/opaque is probably much higher than we previously expected. Thus, and unfortunately, invisible tunnels are much more common than previously stated.

A last question deserves attention: the amount of implicit tunnels being larger than the explicit one, is our uturn heuristic reliable? Indeed, this sub-mechanism is prevalent in our implicit tunnel detection. We can observe that the proportion of signature $< 255, * >$ is less prevalent in implicit tunnels (compared to other categories) and the proportion of $< 64, 64 >$ is also a bit higher (tunnels that do no implement RFC4950 are likely to be older and less mainstream). For the first fact, the reason is obvious: uturn signatures cannot result from such a $< 255, * >$ pattern. The remaining $\approx 5\%$ only comes from the qTTL technique (that is reliable by definition: it does not result from a probing heuristic). Except for these specific signatures, the distributions for implicit and explicit tunnels are relatively close. Such results tend to show that our probing heuristic to detect implicit tunnels seems quite reliable. However, the slight divergences may be due to uturn signatures that are by definition more subject to false positives than qTTL ones.

In order to understand if it is the case, Fig. 6(b) focuses on the implicit tunnels signatures to distinguish our two heuristics. The signature $< 255, * >$ only exists with the qTTL technique. The relative populations of signatures $< 255, 64 >$ and $< 255, 255 >$ balance this decrease for uturn tunnels. It confirms the robustness of our uturn technique: quantity of $< 64, 64 >$ does not move while the natural decrease of $< 255, * >$ is reported to $< 255, 64 >$ and $< 255, 255 >$ classes in the same proportion as for MPLS explicit tunnels. We can conclude that uturn is not the cause of the previous and single $< 64, 64 >$ actual difference (that seems to be induced by the RFC4950 implementation).

4. RELATED WORK

The remote identification of operating systems, also known as OS fingerprinting, aims at discovering the remote machine OS. Based on how data is acquired from the remote machine, two families of OS fingerprinting techniques are possible: *active* (that requires sending traffic towards the target) [22, 23, 24] and *passive* methods (that requires listening to communications between the target and a third-party) [25, 26]. Typically, both families investigate several fields of packet headers. In particular, it focuses on the IP and TCP headers [22, 24], or the various types of ICMP packets [23]. To the best of our knowledge, none of those solutions have explored iTTL n-tuple and applied it to router-level topology discovery.

Closer to our work, Sherry et al. [27] performs alias resolution based on signatures from IP timestamp behavior. Also, Madhyastha et al. [28] use the iTTL in order to estimate the number of hops on the reverse path back from every router to the measurement point. None of them explore iTTL n-tuples nor use signatures to determine possible measurement biases.

Recently, the deployment of MPLS started to be an active research subject. For instance, Sherwood et al. [29] investigated the presence of anonymous and hidden routers as part of DisCarte using signatures based on the IP record route option. Sommers et al. [9] examined the characteristics of MPLS tunnels that are explicitly identified using RFC4950 extensions to statistically infer non explicit ones. In the same vein, we proposed a practical taxonomy of MPLS tunnels based on RFC4950 extensions and `ttl-propagate` option [10]. We have developed techniques for revealing the presence of implicit and opaque tunnels. As demonstrated on this paper, our TTL-based fingerprinting method can be used to refine MPLS identification and quantification.

5. CONCLUSION

Router fingerprinting may help for many purposes such as detecting vulnerable routers or abnormal behaviors and alias resolution. In this paper, we proposed a lightweight router fingerprinting technique based on router signature, i.e., a n-tuple made of initial TTL values used by a router when forging ICMP reply packets. We showed that such a signature is suitable to consistently discriminate IP interfaces. Indeed, various router brands and OSes use different deterministic initial TTL values depending on the type of packet to forge.

Based on data collected during a large-scale measurement campaign, we analyzed the mapping of router OS distribution according to router signatures. As a proof of concept, we applied it on our previous work about MPLS tunnel classification and validated heuristics for revealing non explicit MPLS tunnels. At the same time, we refined our previous conclusions about the invisible MPLS tunnels quantification. More generally, our method or its extension could be used both to determine if a sample of routers is representative of the Internet router mix, and to determine whether a routing feature is independent of the router type.

As a further work, we envision to extend our basic signature. Adding new fields will enlarge the spectrum of possible classes, making them more discriminant. We will try to keep the probing overhead as low as possible while completing our n-tuple of initial TTLs with additional and possibly orthogonal features. We intend to study, among other features, the ICMP packet size, the LSE-TTL field, and the MPLS label range. In a second step, we would like to develop a new multi-probing `traceroute` tool for inferring equipment-based paths in the Internet. Generally speaking, our method can be used to understand whether IP networks are heterogeneous in terms of hardware and software and for analyzing the new OS deployment and market share evolution at different scales.

Acknowledgments

This work is partially funded by the European Commission funded mPlane ICT-318627 project.

6. REFERENCES

[1] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Nmap Project, 2009, see http://nmap.org/book/toc.html.

[2] T. Kohno, A. Broido, and k. claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, May 2005.

[3] F. Veysset, O. Courtay, and O. Heen, "New tool and technique for remote operating system fingerprinting," April 2002, see http://www.leetupload.com/database/Misc/Papers/remote_os_detection.pdf.

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

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

[6] N. Davis, "Initial TTL values," November 2011, see http://noahdavids.org/self_published/TTL_values.html.

[7] A. Sebastian, "Default time to live (TTL) values," December 2009, see http://www.binbert.com/blog/2009/12/default-time-to-live-ttl-values/.

[8] J. Postel, "Assigned numbers," Internet Engineering Task Force, RFC 1700, October 1994.

[9] J. Sommers, B. Eriksson, and P. Barford, "On the prevalence and characteristics of MPLS deployments in the open Internet," in *ACM SIGCOMM Internet Measurement Conference*, November 2011.

[10] 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.

[11] V. Jacobson et al., "traceroute," UNIX," man page, 1989, see source code: ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[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 SIGCOMM Internet Measurement Conference (IMC)*, October 2006.

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

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

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

[16] L. Jacquin, V. Roca, M. A. Kaafar, F. Schuler, and J. L. Roch, "IBTrack: an ICMP black holes tracker," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, December 2012.

[17] A. Medina, M. Allman, and S. Floyd, "Measuring interactions between transport protocols and middleboxes," in *Proc. ACM SIGCOMM Internet Measurement Conference (IMC)*, October 2004.

[18] M. H. Keio, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, November 2011.

[19] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, October 2013.

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

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

[22] Fyodor, "Remote OS detection via TCP/IP stack fingerprinting," *Phrack*, vol. 8, no. 54, October 1998, see http://nmap.org/nmap-fingerprinting-article.txt.

[23] O. Arkin, "A remote active OS fingerprinting tool using ICMP," *;login: the Magazine of USENIX and Sage*, vol. 27, no. 2, pp. 14–19, October 2002.

[24] J. Padhye and S. Floyd, "Identifying the TCP behavior of web servers," in *Proc. ACM SIGCOMM*, August 2001.

[25] C. Smith and P. Grundl, "Know your enemy: Passive fingerprinting," March 2002, see http://www.linuxvoodoo.com/resources/security/finger.

[26] M. Zalewski, "p0f," see http://lcamtuf.coredump.cx/p0f3/.

[27] J. Sherry, E. Katz-Bassett, M. Pimenova, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy, "Resolving IP aliases with prespecified timestamps," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, November 2010.

[28] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November 2006.

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