Towards a Renewed Alias Resolution with Space Search Reduction and IP Fingerprinting

Jean-François Grailet, Benoit Donnet
Université de Liège, Montefiore Institute – Belgium

***Abstract*—Since the early 2000's, the Internet Topology has been frequently described and modeled from the perspective of routers. To this end, alias resolution mechanisms have been developed in order to aggregate all IP interfaces of a router, collected with traceroute, into a single identifier. So far, many active measurement techniques have been considered, often taking advantage of specific features from network protocols. However, a lot of these methods have seen their efficiency decrease over time due to security reinforcements across the Internet.**

In this paper, we introduce a generic methodology to conduct efficient and scalable alias resolution. It combines the space search reduction of TreeNET (a tool for efficiently discovering subnets) with a fingerprinting process used to assess the feasibility of several state-of-the-art alias resolution methods, using a small, fixed amount of probes. We validate our method along MIDAR on an academic groundtruth and demonstrate that our methodology can achieve similar accuracy while using less probes and discovering subnets in the process. We further evaluate our method with measurements made on PlanetLab towards several distinct ASes of varying sizes and roles in the Internet. The collected data shows that some properties of our fingerprints correlate with each other, hinting some observed profiles could be linked with equipment vendors. Both TreeNET (which implements our methodology) and our dataset are freely available.

I. Introduction

For more than a decade now, the Internet topology discovery has been an extensive research subject [1]. Historically, this topology might be seen at three different levels: IP interfaces, routers, and autonomous systems (ASes). In the IP interfaces graph, nodes refer to interfaces collected by traceroute, while links between nodes are links between IP interfaces. The router graph can be obtained by grouping all interfaces of a given router into a single identifier. This process is known as *alias resolution*. Finally, the AS level is obtained when one looks only at ASes and the links between them (in some sense, one aggregates all routers belonging to a given AS into a single identifier, the AS number). Recent developments have suggested to improve this historical vision by adding the Point-of-Presence (PoP) level (referring to routers grouped by geolocation [2], [3]), or the subnetwork level (a set of devices that are on the same connection medium and can communicate directly with each other at the link layer [4], [5]).

Inferring the router level topology of IP networks is an important concern in particular to study routing characteristics. More specifically, inferring the design of an AS is crucial for analyzing intra-domain routing protocol performance. Network protocols designers could evaluate the performance of their proposals on realistic topologies in order to highlight their advantages and limitations. For example, performance of fast-rerouting schemes or multipath transport protocols may strongly depend on the underlying topology. Inferring the architecture of an AS at the router level may help them to develop efficient solutions able to perform well on various topology designs and common patterns. The accuracy of alias resolution is, thus, of the highest importance as also reported by Gunes and Sarac [6].

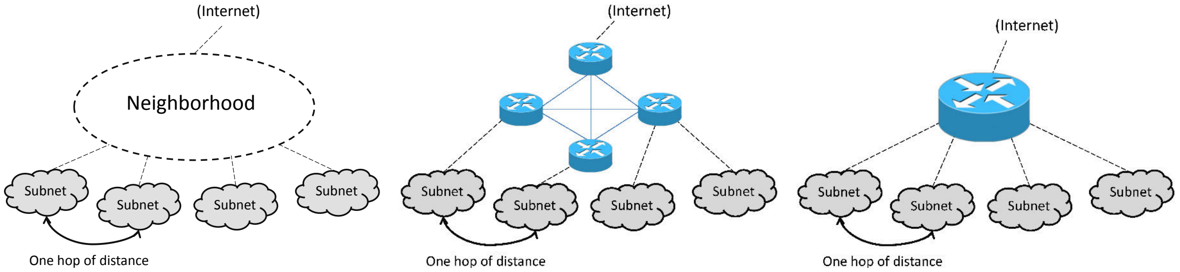
In this paper, we introduce a general methodology to make the best possible usage of state-of-the-art alias resolution techniques. This generic approach, which does not depend on a particular protocol or alias resolution method, combines a *space search reduction technique* (i.e., chunking the alias candidates set into smaller sets) induced by TreeNET [7] (a subnet inference tool, currently only available for IPv4) and a new *fingerprinting* process meant to study the behavior of alias candidates and identify the most suitable alias resolution technique. Fingerprinting, in this context, consists in deriving a vector of values after collecting data on the alias candidates via multiple probes.

We validate our approach by analyzing a groundtruth network with TreeNET, that implements our methodology, and the state-of-the-art tools MIDAR [8] and kapar [9]. Our validation shows that the upgraded TreeNET is able to achieve accuracy close to that of MIDAR while using less probes and discovering subnets. We also evaluate our methodology further by analyzing measurements we performed from the PlanetLab testbed. Study of the fingerprints we obtained supports the conclusion of an early study of fingerprinting that IP interfaces showing a defined behavior can be linked with the hardware brand to which they are assigned [10].

The remainder of this paper is organized as follows: Sec. II provides the required background for this paper by reviewing state-of-the-art alias resolution techniques; Sec. III presents our alias resolution methodology; Sec. IV presents a validation of our methodology along a comparison with state-of-the-art tools on an academic groundtruth, Sec. V evaluates furthermore our method through measurements conducted from the PlanetLab testbed; finally, Sec. VI concludes this paper by summarizing its main achievements.

II. Background

In this section, we review the common approaches for resolving aliases. In particular, we focus on active techniques (i.e., done at the same time as traceroute or shortly after, with additional probing) rather than passive approaches. Such



(a) Conceptual view (b) 1st scenario: mesh of routers (c) 2nd scenario: single router

Fig. 1. The concept of neighborhood.

approaches range from analytical methods like in-depth analysis of `traceroute` records, as implemented in `kapar` [9], to lightweight probing like IGMP probing [11]. The latter has the advantage of silently collecting all multicast interfaces of a router in a single probe. However, as filtering is now heavily applied by ISPs, it is out-dated [12].

First, the *DNS based method* considers similarities in router host names and works when an AS uses a systematic naming scheme for assigning IP addresses to router interfaces. It has the advantage of avoiding direct probing of each router interface. Ally uses this technique against unresponsive routers with the help of the Rocketfuel's DNS decoder [13]. *AROMA* [14] also combines this method and Ally's technique. However, it has been shown that DNS names can introduce errors due to misnaming, leading so to poor alias resolution [15].

Second, the *address based method* is described in RFC 1122 [16]. The principle is simple: the source sends a UDP probe with a high port number to the router interface x. If the source address of the resulting ICMP `Port-unreachable` is y, then x and y are aliases for the same router. The drawback of this solution is that some routers do not generate ICMP messages, making alias resolution impossible. This technique has been implemented in many tools, such as *iffinder* [17] and Mercator [18].

Third, the *IP identifier based method* relies on the IP identifier field of an IPv4 header (or IP-ID), a 16-bit field used to identify the fragments of one datagram from those of another. This field is supposed to be unique for a given (source, destination) pair and protocol. The counter used by a router to choose a value for this field is often the same for all interfaces and it is expected to be simply incremented at each received packet. As a consequence, this field has been exploited for alias resolution by tools like *Ally* [13], *`RadarGun`* [19], and *`MIDAR`* [8]. In particular, `RadarGun` and `MIDAR` study the speed at which the counter increments and alias IP addresses when their respective velocity of incrementing their IP identifier is close (`RadarGun`) or show the same monotonicity (`MIDAR`).

Finally, the IPv4 protocol offers several optional fields which were considered for alias resolution, such as the *record route* feature used by `SideCar` [20]. More recently, the timestamp option with prespecified IP addresses, i.e., *prespecified timestamp*, proved to be useful for resolving aliases [21], [22]. However, nowadays, the majority of deployed network equipment block, for security reasons, all IPv4 packets using options. Such a policy is notably recommended by the IETF since February 2014 [23].

III. Alias Resolution Methodology

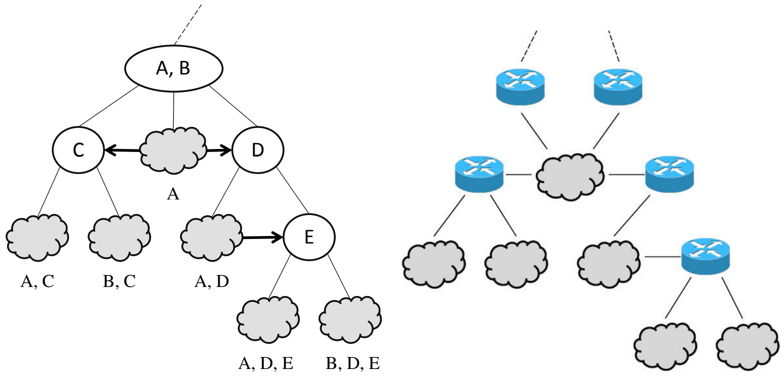
We elaborate a general alias resolution methodology that works in three consecutive steps. First, we use a space search reduction technique to isolate the IP addresses that likely belong to routers and split them in several groups (Sec. III-A). Then, considering one group at a time, we fingerprint each address from a same group (Sec. III-B) to assess the feasibility of different state-of-the-art alias resolution techniques. Finally, we sort the fingerprints and use them to pick the best possible alias resolution method (Sec. III-C).

A. Space Search Reduction

First of all, one should attempt to alias interfaces together only if their respective approximate location in the target domain suggests they could belong to the same device. This idea has already been put to practice by existing tools, such as `APAR` (and its optimized implementation, `kapar`), which considers aliasing interfaces only if their respective distance (expressed as the number of router hops) differ no more than one unit [24].

The first step of our method therefore consists in performing a *space search reduction* (i.e., chunking the set of responsive interfaces to speed-up alias resolution) with `TreeNET` [7]. `TreeNET` is a topology discovery tool that maps a target domain by discovering its subnets and using this knowledge to study the underlying topology. The subnet discovery combines the algorithm of `ExploreNET` [25] with refinement methods introduced by `TreeNET` to evaluate the credibility of subnets and re-construct large subnets which were initially discovered in several chunks. Then, it builds a tree-like structure we called *network tree* to discover *neighborhoods*. A *neighborhood* is a location inside a network bordered by a set of subnets that can all reach each other with at most one router hop. In practice, a neighborhood is either a single router, either a mesh of several routers, possibly connected together with Layer-2 equipment (such as Ethernet switches). Fig. 1 shows an example of a neighborhood, both conceptually and practically, with the two possible scenarios for the real topology.

To discover the neighborhoods of a target domain, `TreeNET` conducts Paris `traceroute` measurements [26]



(a) Network tree

(b) Actual topology

Fig. 2. Example of a network tree and a topology it can model.

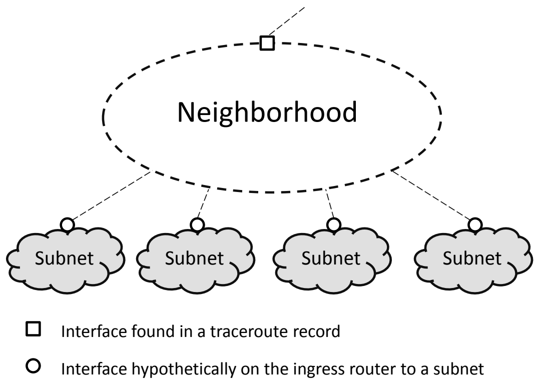

Neighborhood
Subnet
Subnet
Subnet
Subnet
□ Interface found in a traceroute record
○ Interface hypothetically on the ingress router to a subnet

Fig. 3. The *alias candidates* found around a neighborhood.

towards each subnet (i.e., to one responsive IP address of that subnet). Indeed, when two subnets have a route of the same length that ends with the same last hop, they are most likely accessed through the same *ingress router* or the same mesh of routers (with or without Layer-2 equipment), and therefore belong to the same neighborhood. Our *network tree* is essentially an algorithmic approach to gather together subnets for which the route is very similar. The main idea is to view subnets as the tree leaves, while the internal nodes will model neighborhoods. Each internal node bears a *label*, i.e., an IP address found in a `traceroute` record at the position matching the depth at which the node is located. As a set of `traceroute` records consists most of the time of a directed acyclic graph rather than a tree, the structure was adapted to allow internal nodes to bear several labels. These nodes model the fact that, at a given hop count in similar `traceroute` records (i.e., their final hops are almost identical), the observed interface varies from one record to another due to traffic engineering (e.g., load balancing) and can be seen as a superposition of neighborhoods. This modification allows deeper internal nodes to gather all subnets which the route ends with the same last hop, as building a tree without it would result in neighborhoods being split across several branches[1]. Later, we group together child subnets of an internal node with multiples labels when they share the same last hop in their respective route. This simple grouping by last hop allows us to disambiguate an internal node with multiple labels and get back to individual and sounder neighborhoods, rather than a superposition.

Fig. 2(a) shows a toy example of a network tree. Grey clouds model the leaves of the tree, the subnets, while black circles represent internal nodes, therefore neighborhoods. Each subnet is annotated with a possible route to it, while each internal node is labelled with the route hop(s) crossed to reach its children. Notice the black arrows: they show the fact that a subnet can encompass labels of close internal nodes, meaning that the subnet acts as a link between its parent node and the children internal nodes of the same parent node in the actual topology. These arrows are therefore not part of the structure, but rather an observation made after building the tree. For the sake of clarity, we also provide Fig. 2(b) to show a possible actual topology matching our toy example.

What makes the discovery of neighborhoods especially useful in the context of alias resolution is the fact that one can identify, within a subnet, the interface located on its ingress router. Indeed, while most observed interfaces of a subnet are located at the same distance, at least one should be located one hop sooner: the interface located on the ingress router. In certain situations, there might even be several interfaces of that kind, such as back-up interfaces for critical subnets[2]. Discovering neighborhoods therefore amounts to discovering groups of interfaces that necessarily belong to routers and located in a same area, because found either at the last hop in the `traceroute` records of the local subnets, either at a specific amount of hops in the same subnets.

Therefore, during alias resolution, we will only consider for aliasing IP addresses from a same group rather than the whole addresses set which are likely to be router interfaces. Doing so, we expect to spare a lot of effort in the alias resolution process, especially in terms of probing. We will refer to IP interfaces likely to belong to routers as *alias candidates* in subsequent sections. Fig. 3 shows a neighborhood as seen in a network tree and the alias candidates surrounding it.

Starting alias resolution from the observation of a network tree is especially interesting for `TreeNET`, as it allows the tool to naturally extend into a router – subnet topology discovery tool, instead of focusing on only subnets or routers.

B. *Fingerprinting*

The second step of our methodology consists in collecting data from each alias candidate of a same neighborhood and fingerprint them, using multiple probing methods. Each interface is probed as follows: (*i*) several ICMP probes, within a short timeframe, (*ii*) a single UDP packet to a very high port number, and, (*iii*), a single ICMP `timestamp-request`.

[1] Interested readers might refer to Grailet et al. [7] for more details.

[2] Again, we cover this topic with more details in [7].

Additionally, the DNS name of each interface is retrieved when possible.

The ICMP probes, simply consisting in `echo-request`, have multiple purposes. Their primary task is to collect a sequence of IP-IDs via the encapsulating IPv4 headers. The delays (wall clock time, in milliseconds) between consecutive observed IP-IDs are also recorded, and an integer token (unique among all probes) is assigned to each IP-ID to keep track of the order in which probes were sent. Indeed, ICMP probes sent to multiple alias candidates are scheduled to ensure interleaving between tokens. The tokens along the IP-IDs allow us to later use the method applied by Ally (i.e., for two interfaces, it requires four IP-IDs with interleaving tokens), while the delays are useful to estimate the speed at which the IP-IDs of a given IP address increments, in order to alias interfaces with similar speeds when Ally cannot be used.[3] The secondary task of the ICMP probes is to detect when an interface simply echoes the IP-ID in the ICMP `echo-request` packet. Finally, the TTLs found in the replies are also checked to infer what was the initial TTL of the `echo-reply` packet, as it has been previously demonstrated that the initial TTL value is related to a router brand [10]. It simply consists in picking the typical initial value (32, 64, 128, or 255) that is just above the remaining TTL in an `echo-reply` packet. Two alias candidates that do not have the same initial TTL should never be aliased.

The amount of ICMP probes can be configured in `TreeNET`, but by default, we set it to four probes per IP address, for a total of six probes for the whole fingerprinting of a single IP address. This choice is a compromise between being able to evaluate the speed at which IP-IDs increment (with at least three time periods) and using a small amount of probes, to avoid the target domain identifying the probes as an attack.

The collected IP-IDs are also used to derive an *IP-ID counter class*. It is a class derived from the sequence of IP-IDs collected for a given IP address. We consider three classes: *echo*, *healthy*, and *random*. An IP interface will have the *echo* class if it always replies with the same IP-IDs as those sent along the probe packets, i.e., it *echoes* the IP-IDs. On the other hand, the *healthy* class label denotes an IP address which does not echo IP-IDs and for which the IP-IDs form a sound increasing sequence. Finally, an IP address with the *random* class does not echo IP-IDs, but the collected IP-IDs do not form a sound increasing sequence and seem to be drawn at random. This simple classification allows us, later, to quickly check whether IP-ID-based methods are viable or not.

The single UDP probe is sent to obtain an ICMP `Port-unreachable` required for the address-based alias resolution method. Finally, the ICMP `timestamp-request` provides an additional piece of information, i.e., whether this interface replies to such a request or not. Indeed, implementing the ICMP `timestamp-reply` is optional for routers, and the fact that two alias candidates reply to ICMP `timestamp-request` might be an additional hint that they belong, to the least, to a similar device.

(1)	(2)	(3)	(4)	(5)
<255,	w.x.y.z,	Echo,	No,	No>
<255,	*,	Echo,	No,	No>
<64,	*,	Echo,	Yes,	No>
<64,	*,	Healthy,	Yes,	No>
<64,	*,	Healthy,	No,	No>

Fig. 4. Examples of fingerprints. The five values of a fingerprint vector are: inferred initial TTL (1), source address of ICMP `Port-unreachable` (2), IP-ID counter class (3), existence of DNS (4), and compliance to ICMP `timestamp-request` (5).

The fingerprint we derive from the data collected for each interface therefore consists of a vector of five values:

1) the inferred initial TTL of the `echo-reply`,
2) the source address of the ICMP `Port-unreachable`, if the interface was responsive to UDP,
3) the IP-ID counter class,
4) the existence of a DNS (*Yes* or *No*),
5) the implementation of the reply to ICMP `timestamp-request` (*Yes* or *No*).

Fig. 4 shows examples of fingerprints. Whenever part of the data is unavailable, the corresponding part in the fingerprint is set to `*`.

C. Selecting an Alias Resolution Method

The final step of our approach is the actual alias resolution. Just like during the fingerprinting process, we start with the neighborhoods that were previously inferred by `TreeNET`, one neighborhood at a time. The fingerprints computed for a given neighborhood are sorted such that *similar* fingerprints appear consecutively in the list. We consider two fingerprints as similar if each value of the vectors is identical, except for the DNS part, because we observed more than once routers where only specific interfaces had a host name.

Then, we consider groups of similar fingerprints, one at a time, and pick an alias resolution method depending on the available data. If all corresponding IP addresses replied to the UDP probes, we try the address-based approach, which is likely to be the most accurate since another interface is explicitly mentioned in the fingerprint. It should be noted that, if the IP interface that replied to the UDP probe appears in the full list of fingerprints, both addresses will be aliased no matter what the fingerprint of the second IP address looks like. If the address-based approach cannot be used and if fingerprints have the *healthy* IP-ID counter class, we will rather use the IP-ID-based methods, i.e., the same technique as Ally and a velocity-based method if Ally cannot be used (as explained in Sec. III-B). These methods are also used to merge a newly obtained alias with one previously obtained through the address-based approach, if the related fingerprints also had the *healthy* IP-ID counter class.

The last groups of similar fingerprints should be those for which neither the address-based, neither the IP-ID-based approaches can be used. In such a case, we rely on the fact that we drastically reduced the problem to a small set

[3] This method, inspired by `RadarGun` and `MIDAR`, is also detailed in [7].

of alias candidates (through neighborhood inference and the grouping of similar fingerprints) and group the corresponding IP addresses into an alias. There is however an exception: if the host names of two alias candidates are too different (i.e., there are differences beyond the first dot), we do not alias them. In other words, we use reverse DNS not to build aliases but to reject potential aliases. Indeed, building aliases through reverse DNS would require additional inputs to comply with the different naming conventions observed in the target domains and maximize accuracy. We also use this policy for IP interfaces for which the fingerprint is nearly empty (i.e., only the DNS is known). Our approach is implemented in `TreeNET`.

IV. Validation

Before deploying in the wild our alias resolution methodology, we first validate its implementation in `TreeNET` and compare it with other state-of-the-art tools on a groundtruth network. In Sec. IV-A, we briefly describe our groundtruth network and the methodology we followed. Then, in Sec. IV-B, we present and discuss our results.

A. Groundtruth and Methodology

We ran `TreeNET` on an academic network for which we know the actual routers and their respective interfaces.[4] It is important to note that we used a (single) internal vantage point, as a firewall drastically reduces the amount of responsive IP addresses if we probe the same network from an external vantage point.

The groundtruth network is made of one /16 IPv4 block completed with two additional /24 blocks used for the backbone. It is worth noticing that the known topology is essentially in the /16 block and that only a portion of the routers from the backbone is known. Moreover, it should also be noted that most if not all interfaces within the /16 are fit for IP-ID-based alias resolution.

We also ran `MIDAR` on the same network for the sake of comparison. However, while `TreeNET` could be used "*as is*" to probe the network and discover both subnets and aliases, `MIDAR` required some preparation. Indeed, probing all IP interfaces within the target blocks to list potential addresses that should be considered for alias resolution would have made the first step of `MIDAR` needlessly long. We therefore used one of the first steps of `TreeNET`, known as *pre-scanning*, to list all responsive interfaces within the target network and avoid all unresponsive ones. This speeds up `TreeNET` as well as the first step of `MIDAR`.

In addition, we also collected traces with Paris `traceroute` to all responsive IP addresses in order to be able to use `kapar`. Our motivation for using this tool is that it also performs a kind of space search reduction to isolate IP addresses that are likely to be on the same device, as we mentioned in Sec. III-A.

	TreeNET	**MIDAR**	**kapar**
True positive rate	81.78%	98.14%	0.19%
False positive rate	0.22%	0.29%	0.12%
False discovery rate	3.6%	3.65%	91.67%
Precision	96.39%	96.35%	8.33%
Accuracy	98.6%	99.6%	94.47%
Duration (alias reso.)	3'45"	1h47	A few sec.
Duration (total)	1h56	2h28	2h12
# probes (alias reso.)	1948	$\sim 6.6 \times 10^5$	0

TABLE I
Validation results

B. Results and Discussion

Table I shows the main results of our validation, based on the alias pairs obtained by each method. The total duration at the bottom of the table includes all steps mentioned earlier; for instance, the 2h12 total duration of `kapar` is due to the elimination of unresponsive addresses and `traceroute` to responsive ones. Overall, our alias resolution methodology implemented in `TreeNET` shows very good accuracy along with a low false positive rate, though it falls short behind `MIDAR` in terms of overall accuracy. However, looking at the second part of the table shows that `TreeNET` is considerably more economic when it comes to probing. Indeed, during our tests, `MIDAR` sent more than half a million of probes, with an average of 37 probes sent per target interface during its second stage where it estimates the speed of the IP-ID counters of each IP address. `TreeNET`, on the other hand, sends only an average of 6 probes to each address considered for alias resolution and completes the collection of alias resolution hints (see Sec. III-B) in minutes (space search reduction along subnet inference are however included in the total duration).

While `TreeNET` achieves accurate alias resolution much faster, it also comes with a lower true positive rate than `MIDAR`. This true positive rate is a consequence of a higher rate of false negatives rather than inaccuracy of our aliasing methods, and we explain this higher rate of false negatives by the existence of *incomplete* neighborhoods in the network tree built by `TreeNET`. A neighborhood is said to be incomplete when the subnets appearing around it in the data do not include all observable subnets appearing around the same neighborhood in the real network. The main cause of this phenomenon is traffic engineering (e.g., load balancing). Indeed, traffic engineering can cause slight variations in routing: for two subnets which can reach each other with at most one hop in the network, it is indeed possible that one will be reached through a different route than its neighbor, or a slightly longer route (this phenomenon is also known as *route stretching*). Therefore, they will appear in different places during space search reduction, which will prevent the alias resolution from correctly aliasing the interfaces on their common ingress router.

We observed a practical case of this issue on our groundtruth: one subnet featured a unique route when compared to what should be its neighbors in the real network, and produced a separate neighborhood with two alias candidates. Because the real neighborhood was quite large and

[4] For security reasons, we do not provide this groundtruth in our repository.

implemented by a single router, the rate of false negatives among alias pairs was noticeably increased. This observation is confirmed by the fact that, if we manually fix the large alias to add the two missing interfaces and re-run our validation, the true positive rate rises to 85.07% for TreeNET.

We intend to mitigate this issue in the future by adding post-processing steps (with or without additional probing) to both the measurements and the construction of the network tree. In particular, we believe improving our (Paris) traceroute step and post-processing its records could prove useful.

Last but not least, while the space search reduction itself takes time (total execution of TreeNET is around one hour and 56 minutes), it is important to keep in mind that it comes with very useful data that other tools do not provide: subnets. On our groundtruth, TreeNET manages to have more than 90% of its inferred subnets to be faithful to the topology, leading, along with our aliases, to a rather complete map of our groundtruth. Collecting both subnets and aliases at once is very promising for modeling, and future tools could also embed alias resolution in a similar fashion to TreeNET to both collect aliases more easily and obtain a more complete mapping of a network.

Finally, it goes without saying that kapar is not well suited in this situation. Indeed, most if not all its correct aliases were found in the backbone, and moreover, our groundtruth only contains a part of it. This is why the true positive rate is so low. kapar is therefore not suited for studying and modeling the topology of an "end" network such as our groundtruth, unlike TreeNET and MIDAR.

V. Deployment

In this section, we evaluate furthermore our alias resolution methodology from several perspectives. First, we describe how we deployed TreeNET to measure different ASes (Sec. V-A). Then, we analyze the collected data to quantify space search reduction (Sec. V-B) and to discuss the relevancy of fingerprinting for alias resolution (Sec. V-C). Finally, we discuss the overall merit of our methodology with respect to alias resolution (Sec. V-D).

A. Measurement Methodology

We used the BGP Toolkit of Hurricane Electric[5] to select ASes of varying sizes and roles in the Internet topology. We listed 20 different ASes and their respective IPv4 prefixes with an amount of potential addresses ranging from a bit more than 30,000 to a little bit less than 2 millions. To ensure we had different profiles in our list, we used the AS relationships provided by CAIDA [27]. Table II lists all the ASes we probed, along with their respective name, type (i.e., level in the AS hierarchy graph), and amount of potential addresses. For the sake of clarity, we also assign a number to each AS to denote them in our subsequent plots. The list is also split in two, with the first part listing ASes owning large amount of addresses (i.e., more than 500,000).

N.	ASN	Name	Type	#IPs
1	109	Cisco Systems	Stub	1,600,512
2	10010	TOKAI Com.	Transit	1,860,096
3	224	UNINETT	Stub	1,115,392
4	2764	AAPT Limited	Transit	1,074,688
5	5400	British Telecom	Transit	1,385,472
6	5511	Orange S.A.	Transit	922,880
7	6453	TATA Com.	Tier-1	966,144
8	703	Verizon Business	Transit	873,728
9	8220	COLT Tech.	Transit	1,372,160
10	8928	Interoute Com.	Transit	841,728
11	12956	Telefonica Int.	Tier-1	215,040
12	13789	Internap Net.	Transit	106,240
13	14	U. Columbia	Stub	339,968
14	22652	Fibrenoire, Inc.	Transit	76,544
15	30781	Jaguar Network	Transit	45,824
16	37	U. Maryland	Stub	140,544
17	4711	INET Inc.	Stub	34,816
18	50673	Serverius Hold.	Transit	65,280
19	52	U. California	Stub	328,960
20	802	U. York	Stub	75,264

TABLE II
Target ASes of our campaign

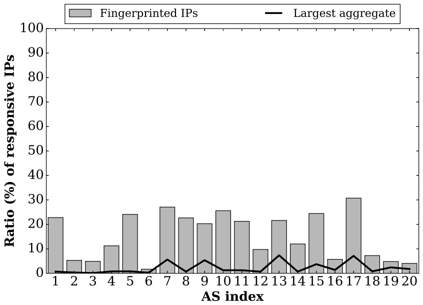


Fig. 5. Benefits of space search reduction (April 14th, 2017).

For each target AS, we ran TreeNET, implementing our alias resolution methodology, on a distinct PlanetLab node (i.e., we used a total of 20 PlanetLab nodes). We renewed our measurements by running periodic campaigns (e.g., we conducted one in January 2017, another started in April of the same year, etc.) during which we probed each AS several times, letting a delay of approximately one day between each consecutive measurement. We chose this delay to avoid imposing a heavy load on the targeted ASes, therefore avoiding any form of blacklisting. The data collected include subnets inferred by TreeNET, the obtained aliases, and all the fingerprints computed during the measurements. The aliases lists also contain the IP addresses which were considered during our alias resolution but could not be aliased at all.

In subsequent sections, we will provide results for the data collected on April 14th, 2017. Those results are typical of what we observed, though a few variations can be seen between consecutive campaigns due to adjustments brought to TreeNET (e.g., improved IP-ID collection scheduling changed the rate of aliases obtained through IP-ID-based methods in April 2017). Nevertheless, interested readers can access our full public dataset on GitHub[6], along with TreeNET.[7]

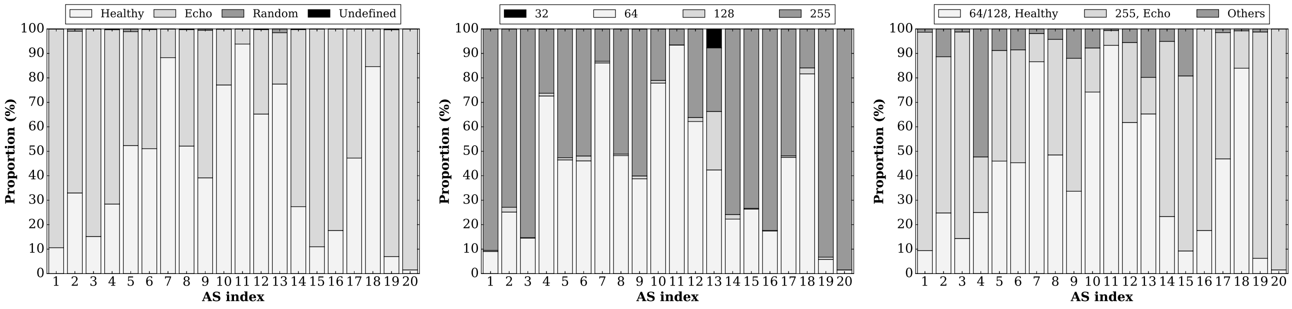
B. Space Search Reduction

We first evaluate the benefits of our space search reduction technique. Fig. 5 shows the percentage of fingerprinted IP

[5] See http://bgp.he.net

[6] https://github.com/JefGrailet/treenet/tree/master/v3/Measurements

[7] https://github.com/JefGrailet/treenet



(a) Spread of IP-ID counter classes

(b) Spread of `echo-reply` initial TTLs

(c) Observed correlation

Fig. 6. Properties of collected fingerprints (April 14th, 2017).

addresses with respect to the total of responsive interfaces, for each AS. With at most 30.71% of the responsive interfaces entering the fingerprinting process, a lot of probing is avoided. The lowest percentage is found in AS5511 (6), with only 1.67%, which can be explained by the fact that most of the responsive addresses at the time of probing were gathered in large subnets, for which only a few interfaces are considered for alias resolution. For example, in a /24, only one or two IP interfaces are taken into account during the fingerprinting process. The black curve shows the largest aggregate of interfaces that were considered together for alias resolution (i.e., from a same neighborhood) with respect to the total number of responsive addresses. It further shows how `TreeNET` simplifies the problem, despite the presence of five peaks. We believe these peaks are a consequence of heavy usage of routing policies such as load balancing, as it complicates neighborhood inference [7].

C. Fingerprinting

Fig. 6(a) shows the spread of the different IP-ID counter classes among the fingerprints collected for each AS. Similarly, Fig. 6(b) shows the spread of the initial TTL values of ICMP `echo-reply` messages (same date). Comparing both figures shows that an IP interface with an *healthy* counter is very likely to use the initial TTL value 64 (or, more rarely, 128), with a few exceptions, as highlighted at Fig. 6(c). As an early study on fingerprinting [10] hinted that initial TTL values correlate with the equipment brand (in particular, an initial TTL of 255[8] likely originates from Cisco equipment), this suggests that grouping addresses with similar fingerprints amounts to grouping IP interfaces which likely belong to devices from a same vendor, which further demonstrates the soundness of our approach.

UDP probing, on the other hand, has been successful for only a few ASes from time to time. In previous measurements conducted with an early version of our upgraded `TreeNET` in April and May 2016[9], three ASes (AS5400, AS703, and AS8220) had addresses responding to UDP probes, with up to 55,60% for AS703. Such an observation does not show, however, that the other ASes block this probing method, as the lack of reply could be due to some intermediate router filtering out the probes.[10] Indeed, we observed a peak of 20% of aliases obtained through the UDP-based alias resolution on January 9th, 2017 for AS8220 (which was the only AS replying to UDP probes at the time), a peak which vanished from the next datasets after the vantage point for this particular AS was changed. Finally, in April 2017, we observed a significant proportion of alias pairs obtained through this method for AS10010 (almost 30%), an AS for which the approach did not work previously. This shows that using a tool solely based on this method has become highly unrealistic, though the method should be at least tried when possible due to its high accuracy and simplicity.

Finally, we evaluate the relevancy of checking if an interface implements the reply to ICMP `timestamp-request`. We observed in our data that, in some cases (notably AS224, AS6453, AS8220, AS8928, and AS52 in our measurements from April 14th), the proportion of devices implementing this mechanism seems to correlate with the proportion of *echo* IP-ID counters. However, this intuition is not always confirmed by the data: the fingerprints of AS8928 shows, for instance, that 771 of the 1,421 fingerprinted interfaces (i.e., 54.25%) which replied to ICMP `timestamp-request` have a *healthy* IP-ID counter. On the other hand, in the case of AS224, 2,306 of the 2,769 fingerprinted addresses (i.e., 83.27%) which feature an *echo* IP-ID counter indeed provide timestamps when queried with ICMP `timestamp-request`. We leave as future work a deeper study of this mechanism in the context of alias resolution and fingerprinting in general.

D. Alias Resolution

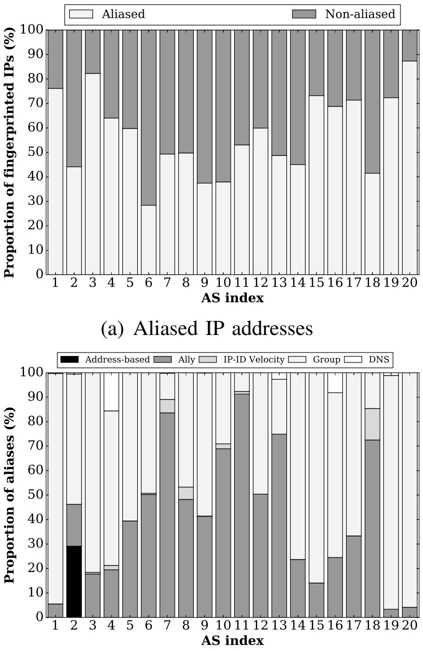
Our alias resolution methodology has the natural advantage of complementing a given technique with another, when the former cannot be applied. In particular, we give priority to the address-based method, when applicable, as it is the only method where a reply is sourced at another interface. In some cases, it can be very useful: almost 30% of alias pairs were obtained with this method on AS10010 on April 14th, 2017.[11] Moreover, since not all interfaces of a router partially discovered through the address-based method will reply to

[8] It is worth noticing that RFC1700 recommends to use 64 as initial TTL value [28].

[9] https://github.com/JefGrailet/treenet/tree/master/v2/Measurements/

[10] Transit filtering has already been observed for IGMP probing [12].

[11] See https://github.com/JefGrailet/treenet/tree/master/v3/Measurements/



(a) Aliased IP addresses

(b) Spread of the different methods

Fig. 7. Alias resolution results (April 14th, 2017).

UDP, we can rely on the IP-ID-based methods (when possible) to build a larger, more accurate alias.

Our approach also has the advantage of providing a coarse-grained alias resolution (see Fig. 5) through the neighborhood inference of `TreeNET`, but also through the grouping of addresses which have similar associated fingerprints. Picking and applying an actual alias resolution technique afterwards therefore acts as a refinement. Furthermore, our approach uses, by default, a fairly low amount of six probes per address for both the fingerprinting and the collection of four IP-IDs for IP-ID-based methods.

Fig. 7(a) shows the proportion of fingerprinted IP addresses aliased with our methodology, for each AS on January 14th, 2017.[12] This shows we can cover large amounts of interfaces while state-of-the-art solutions alone would cover only a subset of them. Finally, Fig. 7(b) shows the proportions of alias pairs obtained through each technique. As expected, the usage of address-based (or UDP-based) method is marginal due to the lack of responsiveness, while IP-ID-based methods are still applicable in a lot of situations. The *Group* bars correspond to situations where no classical method could be used and constrained us to group similar fingerprints from a same neighborhood, because it was the only possible option, while the *DNS* bars correspond to fingerprints which were grouped because there was no new information besides DNS. The considerable size of these bars, once stacked, highlights the importance of performing space search reduction nowadays due to the deprecation of historical approaches.

Of course, our current solution still has room for improvements. For instance, in April 2017, a new and more thorough scheduling for collecting IP-IDs was implemented in `TreeNET` to improve IP-ID-based alias resolution and avoid a maximum of false positives with this approach. In addition to the problem of incomplete neighborhoods mentioned in Sec. IV-B, the way `TreeNET` deals with internal nodes of the network tree bearing multiple labels can also be improved, and we intend to implement in the future a mechanism for identifying the labels which actually belong to a same device to better re-construct individual neighborhoods. Finally, as our DNS-based approach remains very simple, we could also elaborate some heuristics in order to identify the most common naming conventions (e.g., numbered host names) and alias interfaces on that basis.

VI. Conclusion

In this paper, we presented a generic methodology to tackle alias resolution, combining space search reduction and fingerprinting as a way to reduce required additional probing and evaluate the feasibility of different state-of-the-art alias resolution techniques. This combination allows us to consider multiple techniques for small sets of alias candidates, therefore using a fairly low amount of probes.

Using a groundtruth network, we demonstrated that our solution can achieve high accuracy with a reasonable amount of probes while allowing the discovery of other network elements in the process (subnets, in this case). With measurements collected on different ASes from PlanetLab, we also showed that the behavior of fingerprinted IP interfaces could be linked with the vendor of the devices to which they are assigned, which is an additional and useful hint to perform alias resolution.

Of course, our approach can still be improved and deepened. For instance, our space search reduction step remains quite new (to the best of our knowledge) and can evolve in the good direction as topology discovery topics like subnet discovery, in this particular case, are explored. Nevertheless, we believe that our methodology, currently implemented in `TreeNET`, constitutes an adequate response to the current challenges of alias resolution.

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] D. Feldman, Y. Shavitt, and N. Zilberman, "A structural approach for PoP geo-location," *Computer Networks (COMNET)*, vol. 56, no. 3, pp. 1029–1040, February 2012.

[3] Y. Shavitt and N. Zilberman, "Geographical Internet PoP level maps," in *Proc. Traffic Monitoring and Analysis Workshop (TMA)*, March 2012.

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

[5] M. Gunes and K. Sarac, "Inferring subnets in router-level topology collection studies," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2007.

[6] M. H. Gunes and K. Sarac, "Importance of IP alias resolution in sampling Internet topologies," in *Proc. IEEE Global Internet Symposium*, May 2007.

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

[12] Figures for other dates are also available in our repository.

[8] K. Keys, Y. Hyun, M. Luckie, and k. claffy, "Internet-scale IPv4 alias resolution with MIDAR," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 383–399, April 2013.

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

[10] Y. Vanaubel, J.-J. Pansiot, P. Mérindol, and B. Donnet, "Network fingerprinting: TTL-based router signatures," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.

[11] P. Marchetta, P. Mérindol, B. Donnet, A. Pescapé, and J.-J. Pansiot, "Topology discovery at the router level: a new hybrid tool targeting ISP networks," *IEEE Journal on Selected Areas in Communication, Special Issue on Measurement of Internet Topologies*, vol. 29, no. 6, pp. 1776–1787, October 2011.

[12] P. Marchetta, P. Mérindol, B. Donnet, A. Pescapé, and J.-J. Pansiot, "Quantifying and mitigating IGMP filtering in topology discovery," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, December 2012.

[13] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, August 2002.

[14] S. Kim and K. Harfoush, "Efficient estimation of more detailed Internet IP maps," in *Proc. IEEE International Conference on Communications (ICC)*, June 2007.

[15] M. Zhang, Y. Ruan, V. Pai, and J. Rexford, "How DNS misnaming distorts Internet topology mapping," in *Proc. USENIX Annual Technical Conference*, May/June 2006.

[16] R. Braden, "Requirements for internet hosts. communication layers," Internet Engineering Task Force, RFC 1122, October 1989.

[17] K. Keys, "iffinder," a tool for mapping interfaces to routers. See http://www.caida.org/tools/measurement/iffinder/.

[18] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. IEEE INFOCOM*, March 2000.

[19] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's growing pains with velocity modeling," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2008.

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

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

[22] P. Marchetta, V. Persico, and A. Pescapé, "Pythia: Yet another active probing technique for alias resolution," in *ACM CoNEXT*, December 2013.

[23] F. Gont, R. Atkinson, and C. Pignataro, "Recommendations on filtering of IPv4 packets containing IPv4 options," Internet Engineering Task Force, RFC 7126, February 2014.

[24] M. H. Gunes and K. Sarac, "Resolving IP aliases in building traceroute-based Internet maps," *IEEE/ACM Transactions on Networking (ToN)*, vol. 17, no. 6, pp. 1738–1751, December 2009.

[25] M. E. Tozal and K. Sarac, "Subnet level network topology mapping," in *Proc. IEEE International Performance Computing and Communications Conference (IPCCC)*, November 2011.

[26] 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.

[27] The CAIDA UCSD, "The AS relationship," 2013, http://data.caida.org/datasets/as-relationships/.

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