lmproved algorithms for capturing Internet maps

Kevin Vermeulen

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité

Informatique

Ecole LIP6, LINCS

Présentée par

Kevin VERMEULEN

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Improved algorithms for capturing Internet maps.

soutenue le 2 Juin 2020

devant le jury composé de :

M. Prométhée Spathis, Maître de conférences HDR, Sorbonne Université	Directeur de thèse
M. Jim Kurose, Professeur, University of Massachusetts Amherst	Rapporteur
Mme. Cristel Pelsser, Professeur, Université de Strasbourg	Rapporteur
Mme. Anja Feldmann, Professeur, Max-Planck-Institut für Informatik	Examinateur
Mme. Clémence Magnien, Directeur de recherche CNRS, Sorbonne Université	Examinateur
M. Timur Friedman, Maître de conférences, Sorbonne Université	Co-encadrant

Contents

Acknowledgments

First of all, I would like to thank the French Ministry of Defense, for integrally funding my thesis, and all my professional experiences during these three years.
From a professional perspective, I would like to thank: Olivier Fourmaux and Timur Friedman, for accepting me as a doctoral student and for guiding me in my research and my teaching all along the thesis; Prométhée Spathis, for accepting to be my doctoral advisor. Stephen D. Strowes, for our successful collaboration and his invitation to visit RIPE Atlas; Reza Rejaie, for our successful collaboration and his invitation to visit the University of Oregon; Justin P. Rohrer and Robert Beverly, for their incredible kindness in hosting me at Naval Postgraduate School for these three months of fruitful collaboration; Simon Leinen and Niels den Otter, Konstantin Kabassanov, and Ciro Scognamiglio, for letting me running my experiments in SWITCH, SURFNET, LIP6, and PlanetLab Europe networks. network; Niels Den Otter, for Cristel Pelsser and Jim Kurose, for their thorough review of my dissertation; Anja Feldmann and Clémence Magnien for accepting to be members of my thesis committee. Benoit Donnet, and Cristel Pelsser, for inviting me in their universities to give me the possibility to present and improve my work; Marc-Olivier Buob, for his precious help at the beginning of my thesis; My colleagues, Burim Ljuma and Matthieu Gouel, for sharing my work daily life;
From a personal perspective, I would like to thank: Ophelie Walrick, my girlfriend, for being an amazing person, and supporting me every day. You changed my vision and the goals of my life; My parents, for being the most supportive parents. This is easy to say, but hard to reproduce: thanks for adopting me from Vietnam and supporting all of my choices since I was born; My friends, for being the balance that I need between my professional and personal life.

Abstract

English version

The Internet is nowadays a major component of our lives. Also called the network of the networks, the Internet is a giant network composed of thousands of interconnected networks. These networks are composed of routers, themselves having multiple interfaces, that connect to each other to be able to carry the traffic from one point to another. All these connections form a structure, called the Internet topology, that has been questioning the scientific community since its creation. A better knowledge of the Internet topology helps to answer questions such as: Which countries the traffic is crossing? Is the Internet robust to outages? These questions implicitly concern billions of users, and motivate this work to better capture the Internet topology. Network topology being often considered as an industrial secret, researchers perform measurements from vantage points connected to the Internet to reveal it.

This thesis presents new measurement techniques that provide a more complete view of the Internet topology. At IP level, it presents the last evolution of the well known `traceroute` tool and the first system capable of mapping all the IPv4 load balanced paths in 2.5 days, providing the most complete view of the IP level topology from a single server ever obtained. At router level, it presents the first alias resolution technique based on ICMP rate limiting.

This thesis will allow more thorough studies of the Internet through the completeness of the topology discovered, and open the possibility to better study the dynamics of the Internet topology.

French version

Internet joue aujourd'hui un rôle crucial dans notre vie quotidienne. Aussi appelé "réseaux des réseaux", Internet est un réseau géant composé de milliers de réseaux interconnectés. Ces réseaux sont composés de routeurs, eux-mêmes possédant de multiples interfaces, qui s'interconnectent pour transporter le traffic d'un point à un autre. Toutes ces interconnections forment une structure appelée topologie d'Internet, qui intéresse la communauté scientifique depuis sa création. Une meilleure compréhension de la topologie d'Internet permet de répondre à des questions comme: Quels pays mon traffic traverse-t-il? Est-ce qu'Internet est robuste en cas de panne? Ces questions concernent implicitement des milliards d'utilisateurs, et motivent cette thèse pour mieux capturer la topologie d'Internet. Les informations topologiques relevant souvent du secret industriel, les chercheurs effectuent des mesures depuis des points connectés à Internet pour les révéler.

Cette thèse présente de nouvelles techniques de mesures qui permettent une meilleure vue de la topologie d'Internet. Au niveau IP, celle-ci présente la dernière évolution de `traceroute`, et le premier système capable de cartographier les chemins à répartition de charge sur l'ensemble d'IPv4 en 2,5 jours, fournissant la vue la plus complète d'un serveur jamais obtenue. Au niveau routeur, celle-ci présente la première technique de résolution d'alias basée sur le "ICMP rate limiting".

Cette thèse permettra des études plus exhaustives d'Internet, grâce à la complétude des topologies découvertes, ainsi que des perspectives d'études sur les dynamiques de la topologie d'Internet.

Chapter 1

Introduction

Internet, the "network of networks", has been questioning researchers since its creation. In particular, since Internet has expanded to thousands of interconnected networks, and millions of interconnected machines, revealing its structure, or its topology, is a challenge that have been of primary interest. Despite all the efforts of the scientific community; there have beeen hundreds of papers published on Internet topology since 1990 ([50, 69, 134, 34, 65] to only cite those); no one has a clear and exhaustive view of the real Internet topology [32, 117, 94]. At a high level, the Internet is composed of Autonomous Systems (AS) communicating together. According to the RFC 1930 [76], "an AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy." Each AS has a knowledge of its own networks, and does not share this knowledge with others; networks topology is often considered as an industrial secret. Moreover, the Internet topology is a dynamic structure. Outages, attacks, or new peering agreements between autonomous systems are some of the events that can temporarily, or definitively, modify the Internet topology.

In this introduction, we first motivate the needs for capturing Internet topology (Sec. 1.1), then present the different levels of the Internet topology (Sec. 1.2), and present high level metrics to better capture Internet topology (Sec. 1.3). We finally introduce our different contributions to the field (Sec. 1.4).

1.1 Motivation

Internet topology is crucial to our understanding of the Internet because it supports a lot of other works that have a direct impact on the end user of the Internet. Supported topics include resilience to outages or attacks [122, 92], security [133], troubleshoot of congestion [90], geolocation [84], or also mapping of Autonomous Systems interconnections [104].

Outages, attacks, security and congestion are directly related to the quality of experience perceived by the end user, while geolocation involves political issues, e.g., enables Internet censorship [63]. A better mapping of Autonomous Systems interconnections involves socio-economic issues, as Internet providers have an important interest to know with whom their competitors are connected with.

Last but not least, capturing the Internet topology is a challenge in itself for basic science. It is somewhat disconcerting that one of the major advance of the 20th century

remains obscure.

1.2 Multiple levels of the Internet topology

The Internet topology has been studied at three different levels, from coarse grain to fine grain: Autonomous Systems (AS), router, and IP. The Internet is composed of thousands of AS that are connected between each other [71, 97]. The map of these ASes is called the *AS level* of the Internet topology. These AS are physically connected via routers. Each AS is composed of multiple routers that are also interconnected. The map of these routers is called the *router level* of the Internet topology. Each router owns multiple interfaces, that physically or virtually connects with other interfaces of other routers. Each interface is given an IP address and those interfaces communicate via the IP protocol. The map of these interfaces is called the *IP level* of the Internet topology.

A classic approach to build the Internet topology is to first collect the IP level, and then build the router level based on this IP level [3]; if the `traceroute` mechanism can reveal the IP level topology, there is no such "straightforward" technique to reveal the router level. The process to step from the IP level topology to the router level topology is called alias resolution. The AS level can be obtained by jointly using BGP (the Internet protocol that allows communications between AS) [114] data with IP and router level topology [95, 103].

All the three levels are independent fields of research, and in this thesis, we focus on improving the techniques for collecting IP level topology and building router level topology; we provide new techniques for capturing the IP level topology and alias resolution.

1.3 Completeness and accuracy

There are two aspects that can be used to measure the quality of a captured IP or router level topology: completeness and accuracy.

Completeness consists in trying to reveal the most possible components of the Internet topology. This translates into the number of elements that are seen, and the different connections between them. At the IP level, elements are IP interfaces, while at the router level, elements are routers. The more complete, the better.

Accuracy consists in trying to reveal only "real" components of the Internet topology. For example, at the IP level, we do not want to reveal false links [38]. At the router level, we do not want to map two IP addresses belonging to two different routers to the same router, and vice versa, we do not want to map two IP addresses belonging to a same router to two different ones.

These concepts of completeness and accuracy are the unifying goal of this thesis, that is to provide more complete and more accurate maps of the Internet.

1.4 Contributions

We make three major contributions in this thesis that improve the capture of the Internet topology at the IP and the router level, that compose the chapters of the thesis:

Multilevel MDA-Lite Paris Traceroute (MMLPT) [129] (Chapter 3), which improves the accuracy and the completeness of IP and router level topology between a source and a destination.

Diamond-Miner: Comprehensive Discovery of the Internet's Topology Diamonds [128] (Chapter 4), which improves the completeness of IP level topology at Internet scale.

Alias resolution based on ICMP rate limiting [128] (Chapter 5), which improves the completeness and the accuracy of the router level topology at a scale of thousands of IP addresses.

Before delving into the details of each of those contributions, Chapter 2 presents the background and related work of the different contributions necessary to a full understanding of our different pieces of work.

Chapter 2

Related work

This chapter introduces the concepts, background and related work that are fundamentally related to Internet cartography. This includes concepts of traceroute, load balancing, multipath, alias resolution, stateless probing, and Internet mapping systems. Other concepts, background and related work specific to a particular chapter but not necessarily related to Internet cartography are given in the corresponding chapter. We make this choice because we believe this related work will be better appreciated with more context presented in the chapters. This includes network simulators (Chapter 3), Internet path dynamics (Chapter 4), and ICMP rate limiting (Chapter 5).

This chapter follows the structure of the thesis, each of the sections corresponding to one chapter. We finish by summarizing the current challenges of the field that we tackle in the conclusion of this chapter.

2.1 A better multipath traceroute

In chapter 3, we present Multilevel MDA-Lite Paris Traceroute (MMPLT), the last successor of the well known `traceroute` tool, invented by Van Jacobson in 1988 [78]. This section provides the related work on the historical techniques on which MMLPT is built (Sec. 2.1.1), and on IP and router level topology discovery techniques (Sec. 2.1.2).

2.1.1 The lineage of traceroute

Traceroute

Invented in 1988 by Van Jacobson, `traceroute` was firstly designed for network operators for troubleshooting and debugging purposes. It is a command line tool, taking as input an IP destination and giving in output a sequential list of IP interfaces belonging to routers on the path. It leverages the ICMP (Internet Control Message Protocol) [22], the Internet protocol that allows routers or end-hosts to send error and informational messages. In particular, it triggers ICMP time-to-live-exceeded errors (Time Exceeded) from routers on the path to the destination [22]; when a packet enters into a router, the router decrements the TTL header field value of the packet. If this value is decremented to 0, an ICMP time-to-live-exceeded message is sent to the sender of the packet. Note that there exists some cases where the router does not decrement the TTL, in which case the router is called stealth, this can be noticeably encountered in tunnels [66]. `traceroute`

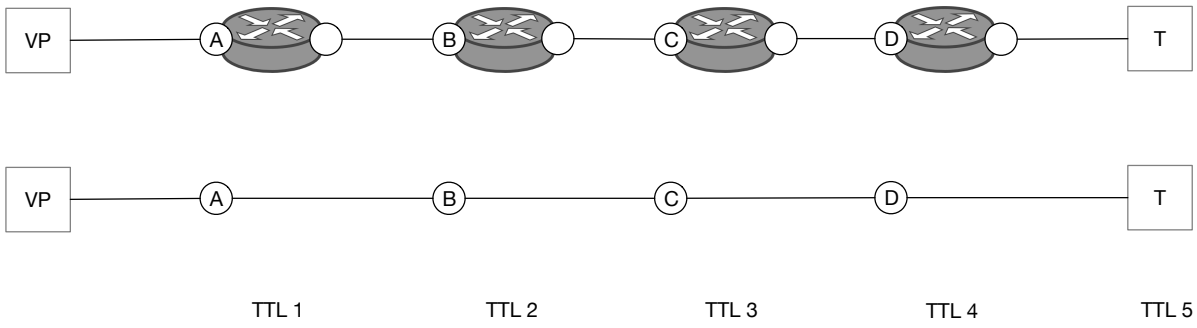


Figure 2.1: Path from VP to T, router level topology (above) and traceroute view (below).

sends TTL limited packets, starting at 1, incrementing the TTL progressively as it receives ICMP Time Exceeded messages from routers. `traceroute` sends different types of packets depending on the protocol. The most standard are UDP packets with high port (starting at 33434 on modern Linux implementations [25]), TCP packets on port 80, and ICMP echo request. When a probe packet reaches the destination, the destination responds with a different type of packet depending on the port and the protocol of the packet: an ICMP destination unreachable error (UDP), a TCP RST, TCP RST/ACK, TCP SYN/ACK (TCP), or an ICMP echo reply (ICMP). This mechanism has the nice property of revealing the IP level path to this destination. From that perspective, researchers have started to use `traceroute` to build IP level Internet topology [1]. Fig. 2.1 gives an example of a router level path between a vantage point VP and a destination T, and the IP level path revealed by `traceroute`, if we assume that the routers are responding with their ingress interface, which is the most common case: A, B, C, D and T.

Load balancing

Load balancing is a feature of routers that allows them to dispatch traffic to a particular destination over multiple paths. This is traditionally done via equal-cost multi-path (ECMP) routing, though other strategies exist [11]. Load balancing is used to increase aggregate network capacity and provide redundancy and resilience to failures. Two types of load balancing are configurable on routers [6, 28, 18]: deterministic and non-deterministic. When a packet arrives on a router configured with deterministic load balancing (i.e., per-flow load balancing), and ECMP paths are available to the packet's destination, the router chooses a path by computing a hash over the packet's header fields [51]. This set of fields used to compute the hash is called the flow identifier, and typically includes the protocol, and either the source and destination addresses (per-destination) or the source and destination addresses and ports (per-flow). Other fields can be utilized, depending on the vendor and the OS [6, 28, 18]. Two packets belonging to the same flow are thus sent over the same path, and this helps the performance of transport protocols that react to delayed or out-of-order packets, as well as enabling middleboxes to have visibility into all the packets of a flow.

Non-deterministic is also known as per-packet load balancing. In this configuration, when a packet arrives at a router with multiple equal-cost paths to the destination, the router selects among the paths in a round robin fashion. Augustin et al. [42] and Ver-

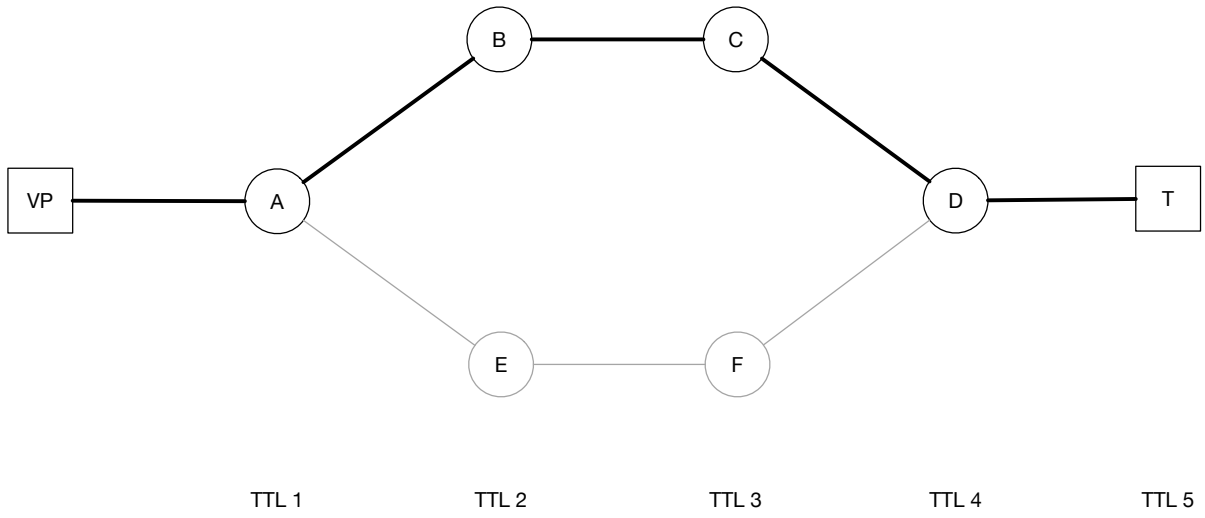


Figure 2.2: IP level topology in presence of a per flow load balancer A and a path revealed by Paris Traceroute in black.

meulen et al. [129, 128] have shown in different surveys that non deterministic load balancing was rare in comparison with deterministic in the Internet (<2%).

If load balancing was not the standard when `traceroute` was created in 1988, with the ongoing increase in capacity, load balancing has become a prevalent practice in configuration of Internet routes. The most recent study from Vermeulen et al. [128] in 2019 revealed that 65% of the Internet routes from a source to a /24 prefix contained load balanced paths.

Paris Traceroute

At its creation, `traceroute` had no reason to take load balancing into account, as load balancing was not a prevalent practice. But as the Internet grew up, load balanced paths became more and more common. And `traceroute`, when used to build IP level Internet topologies, has this flaw of inferring false links in presence of load balancing, as shown by Augustin et al. [38]. Indeed, one of the fundamental mechanisms of `traceroute` is how it matches probes with replies. In UDP and TCP, `traceroute` uses ports: it varies the destination port, which can be retrieved in the ICMP Time Exceeded error payload, which corresponds to the IP header plus 8 bytes of the IP payload of the probe. In ICMP, it uses the sequence number header field of the ICMP echo request probe. This mechanism has a side effect of varying the flow identifier between packets, so that two packets with subsequent TTLs can take two different paths in the presence of per-flow load balancing, leading to false link inference. To overcome this limitation, Augustin et al. designed and implemented Paris Traceroute, an evolution of traceroute which maintains the flow identifier of the probes constant [38]. To be able to still match probes with replies, they used other fields of the header that were not part of the flow identifier. As a result, Paris Traceroute reveals only a "real" IP level path between a source and a destination. Note that this improvement only works with deterministic per-flow load balancing. Fig. 2.2 gives an example of an IP level path in presence of load balancing, with A as an interface

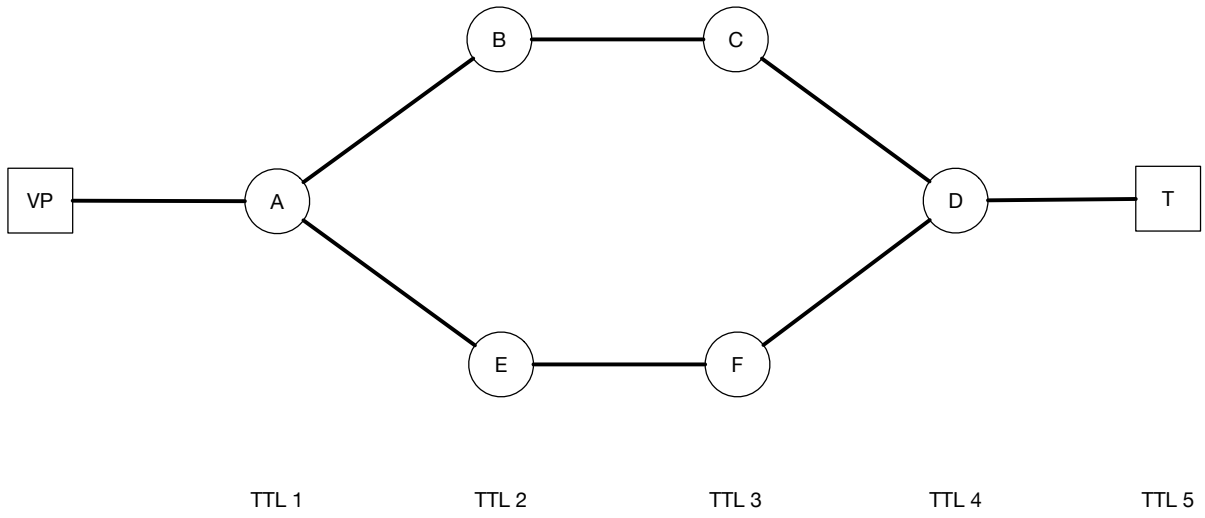


Figure 2.3: IP level topology in presence of a per flow load balancer A and the paths revealed by MDA Paris Traceroute in black.

on a router that is balancing the traffic per flow over B and E. Paris Traceroute ensures that by keeping the flow identifier constant, the probes sent at TTL 2 and 3 will follow the same path, here the upper path. As a result, the IP level path revealed by Paris Traceroute is A, B, C, D, T.

Multipath Detection Algorithm (MDA)

As introduced in the previous paragraph, in the presence of load balancing, there are multiple paths between a source and a destination. Paris Traceroute can be used to reveal one of them, and varying the flow identifier between two instances of Paris Traceroute can reveal multiple paths. However, this *ad hoc* technique does not provide any statistical guarantees about the discovery of these multiple paths. The multipath detection algorithm (MDA) has been designed to model the minimum number of probes to send to achieve a desired level of statistical guarantees to have discovered all the load balanced paths from a particular router. It works at the vertex level; for a given vertex with k known outgoing load-balanced edges, the number of probes with randomly selected flow identifiers needed to verify that it has no more than k edges is denoted n_k, and is termed a "stopping point" [126]. For example, when 1, 10, or 100 outgoing edges have already been identified, $n_1 = 6$, $n_{10} = 57$, or $n_{100} = 757$ probes are, respectively, required in order to ensure a no more than 0.05 probability to fail to enumerate all outgoing edges. Fig. 2.3 shows the same topology as Fig. 2.2 but this time it can discover all the load balanced path between VP and T.

MMLPT is an evolution of the MDA, in the sense that it provides a lightweight version of the MDA, MDA-Lite, which reduces the number of packets sent in the network while preserving the same level of statistical guarantees in presence of a certain type of topologies, that we found to be prevalent in the network.

2.1.2 IP level route tracing

As described in the previous paragraphs, MMLPT builds directly on Paris Traceroute [38, 130] and on the Multipath Detection Algorithm (MDA) [40, 126]. It also inscribes itself in the line of measurement work that has sought to improve our ability to trace the IP level paths that packets take through the Internet, such as Reverse Traceroute [85], which uses the IP Record Route option to learn IP addresses on the return path taken by probe replies; Vanaubel et al.'s Network Fingerprinting technique [125] for examining the TTLs of probe replies to determine which type of router might have sent them and, combined with examination of the MPLS label stack that is received in an ICMP Time Exceeded message, to trace a path's MPLS tunnels; or Dublin Traceroute [43], which uses Steven Bellovin's technique [44] for examining the IP ID field of probe replies for NAT box detection in order to identify NAT boxes on a multipath route. Similarly to the latter two, our multilevel route tracing technique goes beyond the interface level to uncover information about the devices through which packets pass.

2.1.3 Multilevel route tracing

Our multilevel route tracing makes use of existing alias resolution techniques, notably Midar's [87] Monotonic Bounds Test (MBT) for comparing overlapping time series of the IP IDs of probe replies. The MBT itself builds on the pioneering approaches of Ally [120] and RadarGun [45]. We also use Vanaubel et al.'s Network Fingerprinting [125]. These techniques are further described in details in Sec. 2.3.1.

Contribution If MMLPT does not make any contribution to alias resolution in itself (except decreasing the number of probes needed by reusing informations gathered from the probes sent to discover the IP level), it provides the comfort for a network operator to debug a multipath route at router level in a single command line. Prior to MMLPT, one had to first run an IP level discovery technique, and then transform the output into a correct input for the alias resolution technique, which often required additional metadata or installation [87, 93, 91]. MMLPT makes the troubleshooting straightforward by only needing a destination as input.

2.2 Internet Scale multipath tracing

In chapter 4, we present Diamond-Miner, the first system that has the capacity to trace IP level multipaths at Internet scale while providing statistical guarantees. As a topology discovery system, it directly competes with other Internet topology mapping systems. This section provides the background and related work on Internet topology mapping systems.

2.2.1 Internet mapping systems

There are two systems that perform Internet-wide surveys to capture the topology of the Internet: Ark, that is a system developed by Caida, at University of California San Diego (UCSD), and Yarrp in AWS, that is a system developed at Naval Postgraduate School (NPS).

Figure 2.4: Geographic distribution of the Ark Internet topology mapping infrastructure [1].

This section provides details about the characteristics of these systems with respect to Diamond-Miner contributions, i.e., the capacities of the systems to perform at Internet scale and to reveal load-balanced paths..

Ark

Ark [1] is an Internet topology mapping system developed by CAIDA since 2007. It performs continuous surveys of the Internet topology from more than a hundred vantage points ($\approx$ 110 in 2020) geographically distributed around the globe by launching Paris Traceroute measurements to one random address in every globally advertised /24 prefix. Fig. 2.4 shows the locations of the ark nodes around the globe.

A complete set of measurements towards all of the /24 prefixes is called a cycle. In 2019, a cycle was taking a day on average. Because the address in the /24 is randomly chosen, aggregating results from multiple cycles can reveal some multipaths. However, Ark is not explicitly designed to reveal load balancing, does not send enough probes to find all load balancing (particularly when chained), and does not provide any confidence bounds on discovery. Regarding to Diamond-Miner, Ark an Internet scale topology mapping system which does not completely trace load balanced paths.

In addition to this methodology to build an IP level topology of the Internet, CAIDA performs alias resolution from the Ark nodes to build a router level topology from the IP level topology. They provide two datasets of router level topology, obtained with different alias resolution techniques: the first one is obtained with MIDAR [87], and the other is

obtained by performing a transitive closure between MIDAR and KAPAR [12], which are further detailed in Sec. 2.3.1.

Yarrp

Yarrp is a high-speed Internet topology mapping technique that has been developed by Robert Beverly at NPS [46, 47]. It achieves high speed by decoupling the probing from the topology reconstruction. In the standard `traceroute` tool and its successors, as we mentioned in Sec. 2.1.1, the replies are matched with their corresponding probes by varying some values of the fields of the transport header. In particular, the sender maintains a table of which probes have been sent to be able to match the forthcoming replies. This limits the number of probes that can be sent in parallel. With Yarrp, the probes are self identifying, meaning that the TTL and the timestamp at which they were sent are encoded into some IP and transport header fields so that one can retrieve them in the ICMP Time Exceeded error replies that are coming back. As a result, one can reconstruct the traceroutes and the topology *a posteriori*. With high speed probing techniques comes the potential shortcoming of being rate limited by routers. To avoid this phenomenon, Yarrp randomizes the probing, to not overload Internet paths. ICMP rate limiting is further detailed in Chapter 5.

Yarrp has since been deployed on Amazon Web Service cloud in 15 geographically distributed datacenters. The 15 vantage points are achieving a 1 million packets per second probing rate all together [116]. Probing 1 address per /24 such as Ark, Yarrp deployment in AWS is capable of obtaining IP level topology snapshots of the Internet every 15 minutes. Comparing with Diamond-Miner, Yarrp has the same limitation as Ark. It does not discover multipath "on purpose". In the same way, Yarrp can eventually find load balanced paths by by varying the destination per /24 between two snapshots, but it does not provide any statistical guarantees.

2.3 Router level topology

Limited Ltd. is the latest in a long line of alias resolution methods stretching back over twenty-plus years. Alias resolution is the process of grouping IP addresses into routers. By the time of its submission, Limited Ltd. enables alias resolution on Juniper routers in IPv6 for the first time, while being a good complement to state-of-the-art techniques for IPv4. This section situates the contributions of Limited Ltd. with respect to all of the previous work that has been done in alias resolution.

2.3.1 Alias resolution

Alias resolution has been an active area of research for more than two decades now. Table 2.1 presents an inventory of all previously known techniques. The table is organized according to the basis for various groups of alias resolution techniques. This might be a signature, the topology, or something else. A signature-based method works by sending probe packets to IP addresses in order to elicit reply packets that display a feature that is distinctive enough to allow replies coming from a common router to be matched. A topology-based method works by sending probe packets to sequential hop counts in order to obtain IP addresses, and then reasoning on the basis of the IP-level topology to infer

Year	Basis (**s**) = signature (**t**) = topology (**o**) = other	Algorithms and tools	Condition of applicability	IPv4	IPv6
				($\boldsymbol{\tau}$) = tool ($\boldsymbol{\delta}$) = dataset	
1998 [107]	source IP address (**s**)	Pansiot and Grad [107]	respond with a common IP address in ICMP Destination Unreachable messages	yes	
		Mercator [72]		yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)	
2002 [120]	IP ID (**s**)	*Ally* [120]	send replies with a shared IP ID counter that increases monotonically with each reply	yes ($\boldsymbol{\tau}$)	
		RadarGun [45]		yes ($\boldsymbol{\tau}$)	
		Midar [87]		yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)	
2002 [120]	Reverse DNS (**o**)	*Rocketfuel* [120] Aroma [89]	IP address resolves to a name	yes	yes
2006 [74]	`traceroute` (**t**)	Apar [74]	respond with ICMP Time Exceeded messages	yes	
		kapar [86]		yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)	
2010 [118]	IP Prespecified Timestamp option (**s**)	Sherry et al. [118] *Pythia* [101]	fill in timestamps as specified by the option	yes	
2010 [112]	IPv6 source routing (**s**)	Qian et al. [112, 111]	source routing must be enabled		yes
2013 [93]	IPv6 fragmentation identifier (**s**)	*Speedtrap* [93]	IDs elicited from responses increase monotonically		yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)
2013 [119]	IP Record Route option (**t**)	*DisCarte* [119]	fill in IP addresses as specified by the option	yes	
2015 [106]	IPv6 unused address (**s**)	Padmanabhan et al. [106]	126 prefixes on a point to point link		yes
2019 [96]	Reverse DNS regular expression (**o**)	*Hoiho* [96]	IP addresses resolves to a name and a regular expression uniquely identifies the reverse DNS name of a router	yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)	yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)
2020 [102]	Return TTL (**t**)	*APPLE* [102]	respond with ICMP Time Exceeded messages	yes	yes
2020 [127]	ICMP rate limiting (**s**)	*Limited Ltd. [127]*	ICMP rate limiting shared by interfaces of the router	yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)	yes ($\boldsymbol{\tau}$) ($\boldsymbol{\delta}$)

Table 2.1: Alias resolution methods

the router-level topology. The other methods (for the moment, there is just one) do not involve probing the IP addresses in question at all, but rather consult other sources (in this case, the domain name system) and reason on the results.

Note that, if we deviate from the distinction between "fingerprint" and "analytical" techniques from previous literature [74, 86], it is because all methods employ analysis. The term "analytical" originally applied to APAR [74], as it is capable of working on pre-collected topology traces without new probing. But nothing about topology-based methods restricts them to offline retrospective analysis.

The bases for alias resolution are listed chronologically, according to the year in which each was first proposed, starting in 1998 with Pansiot and Grad [107] making use of the tendency of some routers to send probe replies with a common IP source address regardless of the IP address that was probed.

There may be more than one algorithm that makes use of the same basis for alias resolution. Mercator [72], for example, refined Pansiot and Grad's technique.

The condition of applicability designates the prerequisites that must be met for alias resolution to be carried out on the basis in question. For example, to conduct alias resolution on the basis of IP IDs, there must be an IP ID counter that is shared across multiple IP addresses of a node and, except for the most basic technique (Ally [120]), that counter must increase monotonically with each reply.

The table indicates if the basis in question applies to IPv4 and/or to IPv6. If there is currently a publicly available tool, this is indicated. If the tool is used to produce publicly available datasets, this is also indicated.

By the time of the submission of the first paper on Limited Ltd. (October 2019), we made the observation that of all of the IPv6 techniques, there was a publicly-available tool for only one: Speedtrap [93]. Speedtrap has a known limitation of only working on routers that generate monotonically increasing IPv6 fragmentation IDs, whereas there is an entire class of routers, such as those from Juniper, that do not generate IDs this way.

Relying upon monotonically increasing IP IDs for IPv4, as does state-of-the-art MIDAR [87], presents a different issue: fewer and fewer routers treat IPv4 IP IDs this way due to a potential vulnerability [68, 20].

Contribution In this thesis, we propose Limited Ltd. [127], a publicly-available tool that does not rely upon monotonically increasing IDs, thereby enabling IPv6 alias resolution on Juniper routers for the first time and IPv4 alias resolution on a growing class of routers for which MIDAR will no longer work.

Notice that since the submission of the first paper on Limited Ltd. , two more works on alias resolution have been published, APPLE [102] and Hohio [96], both working in IPv6, of which one has an available tool, Hohio. This changes the picture of alias resolution state-of-the-art in IPv6, because Hohio uses reverse DNS names, and is therefore independent of a particular implementation or configuration of a router, i.e., it can work on Juniper routers in IPv6. However, as stated by their authors and by their available validation of their results in [10], Hohio is not a high-precision technique such as MIDAR, Speedtrap or Limited Ltd..

2.4 Challenges

We conclude this chapter by summarizing the challenges in Internet cartography that our thesis answers.

At the IP level, we focus our work on IPv4: there has been ten years since the last study on load balancing characteristics and prevalence [42]. Even at that time, Augustin et al. found that load balancing was already prevalent. An update of load balancing in 2020 Internet is needed, and this is a contribution of chapter 3.

As we will see in chapter 3, load balanced paths have remarkably grown in complexity, with up to thousands of edges and hundreds of interfaces between a single source and a single destination. Internet scale topology mapping systems, as they do not have the capacity to capture the load balanced paths in their survey, miss an important amount of the topology. To provide Internet maps while taking into account this new complexity, a system that captures load balanced paths at Internet scale is needed. This is the main contribution of chapter 4.

Finally, at the router level, the alias resolution area of research is mature for IPv4 but a lot less complete in IPv6. New alias resolution techniques for IPv6 are needed. Also, although the field is mature in IPv4, existing techniques are seeing their applicability declining [68, 20], so fresh ideas for alias resolution for IPv4 are welcome as well. This is the contribution of chapter 5.

Chapter 3

A better multipath traceroute

This chapter presents Multilevel MDA-Lite Paris Traceroute, our first major contribution. This work is motivated by two purposes: (1) to provide a new tool to operators to better troubleshoot their network, (2) to minimize the number of packets that have to be sent in multipath tracing.

This work has been published in the proceedings of the *ACM SIGCOMM Internet Measurement Conference (IMC) 2018*.

3.1 Introduction

Since its introduction by Van Jacobson in 1988 [78], Traceroute has become ubiquitous on both end-systems and routers for tracing forward paths through the Internet between source and destination at the IP level. Network operators use it for troubleshooting; the network measurement community uses it in its studies; and vast numbers of route traces are executed daily by long term Internet survey infrastructure such as Ark [1], M-Lab [16, 67], and RIPE Atlas [23, 115]. Two updates were proposed to Traceroute in 2006-2007 to take into account the ever-increasing presence of load balancing routers: the Paris technique [38, 130], for tracing a single clean path through load balancers, and the Multipath Detection Algorithm (MDA) [40, 126], for discovering all of the load balanced paths at the IP level between source and destination. Well over a billion route traces using the MDA have been executed by M-Lab [52] in the intervening years, and the Paris technique is used for route tracing on the over 10,000 RIPE Atlas probes, and by Ark, the Internet mapping system.

We make four contributions that advance the state of the art for multipath route tracing. First is *MDA-Lite* (Sec. 3.2), a lower overhead alternative to the MDA that is tailored to the most common load balanced topologies that we encounter in the Internet. We identify a characteristic that we call "diamond uniformity" that often holds and that can permit significant probe savings. Second is *Fakeroute* (Sec. 3.4), which validates, to a high degree of confidence, that a software tool's implementation of its multipath route detection algorithm performs as intended on a variety of simulated test topologies. Third is *Multilevel MDA-Lite Paris Traceroute* (Sec. **??**), which, for the first time, integrates router-level view of multipath routes, into a Traceroute tool. Until now this has only been done by other tools once route tracing is complete. Fourth, we provide *new survey results* (Sec. 3.5) for multipath routing in the Internet, both at the IP level, and at the router

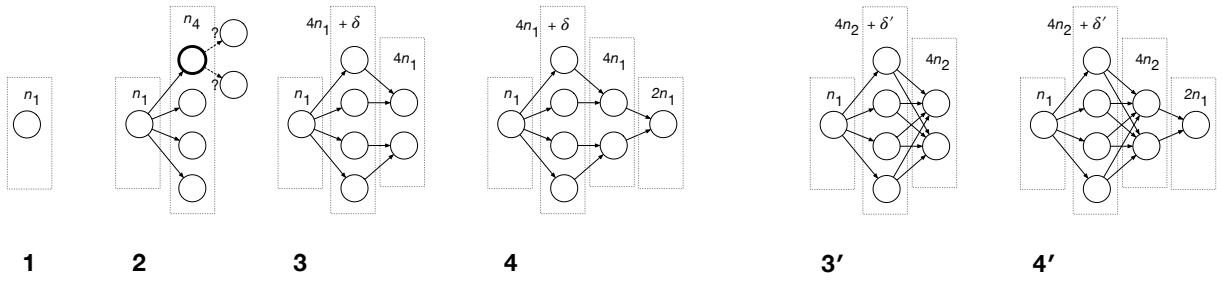


Figure 3.1: MDA discovery of an unmeshed and a meshed diamond

level. We report load balancing practices on a scale (up to 96 interfaces at a single hop) never before described.

3.2 MDA-Lite

The idea behind the MDA-Lite is that we can take advantage of prior knowledge of what a route trace is likely to encounter in order to probe more efficiently. Experience tells us, and our survey in Sec. 3.5.1 confirms, that some multipath route patterns are frequently encountered in the Internet, whereas others are not. The MDA-Lite algorithm operates on the assumption that a topological feature that we call "uniformity" will be prevalent and that another feature that we call "meshing" will be uncommon. It includes tests to detect deviations from these assumptions. We detail these two topological features in Sec. 3.2.2.

3.2.1 The MDA and possible probe savings

This section recalls how the MDA works, stepping us through examples of the discovery of what are called "diamonds", as shown in Fig. 3.1. We see how a feature of the algorithm that we dub "node control" requires large numbers of probes to be sent.

The MDA has evolved through 2006 and 2007 poster and workshop versions [39, 41] to its present form in an Infocom 2009 paper [126]. This latter publication describes an idealized formal model for multipath route discovery [126, Sec. II.A], based upon a set of assumptions about the Internet, and explains the adaptations made [126, Sec. III.A], in crafting the MDA, to accommodate some divergences assumptions and reality. These assumptions are: "(1) No routing changes during the discovery process. [...] (2) There is no per-packet load balancing. (As a result, we can manipulate a probe packet's flow identifier to cause it to pass through a chosen node.) (3) Load balancing is uniform-at-random across successor nodes. (4) All probes receive a response. (5) The effect of sending one probe packet has no bearing on the result of any subsequent probe. In particular, load balancers act independently."

The MDA works on the basis of an *open set* of vertices [126, Sec. II.A], each of which has been discovered but has not yet had its successor vertices identified. A discovery round consists in choosing a vertex v from the open set and trying to find all of its successors. Where there is no load balancing, v has just one successor, but if v is the responding interface of a load balancing router, there will be two or more possible successors that

can only be identified by stochastic probing. In the case that concerns us, per-flow load balancing, successors are found by varying the flow identifier from one probe packet to the next. An extension [40, Sec. 3.2], that we do not employ here, would allow us to measure per-destination load balancing, the effects of which are identical to routing insofar as a single destination is concerned.

The number of probe packets the MDA sends to discover all successors of a vertex v is governed by a set of predetermined stopping points, designated n_k. If k successors to v have been discovered then the MDA keeps sending probes until either the number of probes equals n_k or an additional successor has been discovered. In the latter case, the new stopping point becomes n_{k+1}. Eventually, one of the stopping points will be reached. The stopping points are set in such a way as to guarantee that the probability of failing to discover all of the successors of a given vertex is bounded. Combined with the assumption of a maximum number of branching points, this implies a bound on the failure to discover an entire topology. The MDA takes as a tunable parameter this global failure probability bound and works backwards to calculate the failure bound on discovering all the successors to a given vertex, which in turn determines the values n_k.

Diamond: As defined by Augustin et al. [42], a *diamond* is "a subgraph delimited by a divergence point followed, two or more hops later, by a convergence point, with the requirement that all flows from source to destination flow through both points". Fig. 3.1 provides examples of the MDA successfully discovering the full topologies of two similar diamonds: each one has a divergence point at hop 1, followed by four vertices at hop 2, two vertices at hop 3, and a convergence point at hop 4. Each vertex represents an IP interface, which is to say that these are IP level graphs, not router-level graphs. The full diamonds are shown at steps 4 and $4'$ in the figure. We call the one at step 4 an "unmeshed" diamond and the one at step $4'$ "meshed", the difference relating to the links between hops 2 and 3. Sec. 3.2.2 provides a formal definition of meshing. Since discovery is identical for hops 1 and 2, we show the first two steps for the unmeshed diamond and do not repeat them for the meshed one. A vertex at hop 2 of the unmeshed diamond is highlighted and two hypothetical successors are shown in order to illustrate "node control", a concept described below.

Hop 1: The MDA sends a probe that discovers the single vertex at hop 1. It continues by sending additional probes to that hop, each with a different flow ID, until it reaches the stopping point of n_1 probes, at which point it rules out the existence of a second vertex at that hop. The annotation shows a total of n_1 probes having been sent to hop 1.

Hop 2: The MDA sends a probe that discovers a vertex at hop 2. As with hop 1, it sends additional probes, each with a different flow ID, but in this example it discovers a second vertex on or before having sent n_1 probes. Thus the limit becomes n_2. Third and fourth vertices are discovered before n_2 and n_3, respectively, are met. When n_4 is reached, no fifth vertex has been found and so the MDA stops scanning this hop.

Node control: When a hop has more than one vertex, the MDA works on the hypothesis that each of these vertices is a potential divergence point with successors that are perhaps reachable only via that vertex. It therefore employs what we dub here *node control*, which ensures that each probe packet that goes to the subsequent hop does so via the chosen vertex.

We have illustrated node control with the highlighted vertex at hop 2, and the hypothesis that it has two successor vertices at hop 3. The MDA needs to identify a minimum

of n_1 flow IDs that bring probes having a TTL of 2 to the highlighted vertex in order to send probes to TTL 3 via that vertex. In order to exercise node control for each of the four vertices at hop 2, a minimum of $4n_1$ probes must be sent to hop 2. Depending upon the specific stopping point values, it can be unlikely or even impossible for the n_4 probes that had initially discovered the vertices at hop 2 to have resulted in at least n_1 of them reaching each of the four vertices. To take a numerical example from Veitch et al.'s Table 1 [126, Sec. III.B], $n_1 = 9$ and $n_4 = 33$. In this case, it is impossible for the 33 probes that were used in hop 2 discovery to yield 9 flow IDs for each of hop 2's four vertices; at least $4 \times 9 = 36$ probes would be required for that. 36 probes are unlikely to be distributed perfectly evenly, so some additional probing is necessary. The annotation at hop 2 is updated in the illustration for hop 3 to indicate that $4n_1 + \delta$ probes have been sent to hop 2, where δ is a non-negative integer.

The node control problem is an instance of the Multiple Coupon Collector's problem, which is described by Newman et al. [105] and more recently by Ferrante et al. [70].

Hop 3: Having generated the flow IDs necessary for node control, the MDA now sends probes to hop 3: n_1 probes via each of the four hop 2 vertices. For the unmeshed topology in this example, only one successor vertex is discovered for each hop 2 vertex. The annotation shows a total of $4n_1$ probes having been sent to hop 3.

Hop 4: The MDA also exercises node control at hop 3 in order to probe hop 4. In this example, since n_1 probes have already reached each hop 3 vertex, no further flow IDs need to be generated. The annotation shows a total of $2n_1$ probes having been sent to hop 4, where the diamond's convergence point is discovered.

A total of $11n_1 + \delta$ probes will have been sent overall to discover this topology. Using the values from Veitch et al., $99 + \delta$ probes will have been required by the MDA. The values from Veitch et al. illustrate the cost of node control: $4n_1 = 36$ probes were sent to hop 3, whereas only $n_2 = 17$ probes were strictly necessary at that hop, and twice as many probes than necessary were sent to hop 4.

Hop 2 node control under meshing: The numbers differ for the meshed diamond starting at the third hop, which we distinguish in Fig. 3.1 with the label $3'$. Each hop 2 vertex has two successors at hop $3'$, as opposed to just one at hop 3. Presuming the MDA discovers the second successor in each case, node control requires additional probes to be sent to hop 2 such that there are at least n_2 flow IDs that reach each vertex at that hop. The annotation shows a total of $4n_2 + \delta'$ probes having been sent to hop 2 for the meshed diamond.

Hop $3'$: As the annotation shows, a total of $4n_2$ probes are sent to hop $3'$. The meshing results in more probes than the $4n_1$ probes sent to hop 3 in the unmeshed diamond.

Hop $4'$: There being only one node at hop $4'$, the annotation shows a total of $2n_1$ robes are sent to that hop, just as for hop 4 in the unmeshed diamond.

A total of $8n_2 + 3n_1 + \delta'$ probes will have been sent overall to discover the meshed topology. Using the values from Veitch et al., $163 + \delta'$ probes will have been required by the MDA. Again, we see the cost of node control, here accentuated by the multiple successors to each hop 2 vertex.

Per-packet load balancing: Since per-packet load balancing was found to be rare in Augustin et al.'s 2011 survey [42], we consider that the assumption (2) of no per-packet load balancing described at the start of this subsection is a reasonably good one, and we have omitted the additional packets to check for per-packet load balancing from our

implementation of the MDA, as well as from the MDA-Lite.

3.2.2 Uniformity and meshing

As we see in the Fig. 3.1 examples, the MDA's use of node control is costly in the number of probes that it requires. However, node control is only necessary for certain kinds of diamonds, which we describe here. If diamonds that require node control are sufficiently rare, an "MDA-Lite" could do away with much of the need for node control. As we shall see, a small degree of node control is still required in order to determine which sort of diamond has been encountered. When necessary, the MDA-Lite can switch over to the MDA with full node control.

We have identified a diamond feature that we call "uniformity" that allows full topology discovery without node control. We have also identified a characteristic of diamonds that we call "meshing" that counteracts the potential for probe savings that uniformity otherwise offers. We define uniformity and meshing here and, as we show in Sec. 3.5.1, uniform unmeshed diamonds are indeed very common. Therefore, probe savings can be realized by using the MDA-Lite.

Uniformity: We define a *uniform hop* as one at which there is an equal probability for each of its vertices to be reached by a probe with that hop's TTL and a randomly chosen flow identifier. For a uniform hop, the failure probability bounds associated with the MDA's stopping points, the values n_k, apply to discovery of all the vertices at a hop, and node control is not required. A diamond as a whole is considered a *uniform diamond* if all of its hops are uniform.

Meshing: As already implied, meshing has to do with the links between adjacent hops. Consider hops at TTLs i and $i+1$. We define these to be *meshed hops* if one of the three following conditions applies:

- The hops have identical numbers of vertices and the out-degree of at least one of the vertices at hop i is two or more. Equivalently, the in-degree of at least one of the vertices at hop $i+1$ is two or more.
- Hop i has fewer vertices than hop $i+1$ and the in-degree of at least one of the vertices at hop $i+1$ is two or more.
- Hop i has more vertices than hop $i+1$ and the out-degree of at least one of the vertices at hop i is two or more.

We define a *meshed diamond* as a diamond with at least one pair of meshed hops. The right-hand side of Fig. 3.6 illustrates a meshed diamond, in which hop pairs $(2, 3)$ and $(4, 5)$ are meshed.

3.2.3 The MDA-Lite algorithm

The MDA proceeds vertex by vertex, employing node control to seek the successors to each vertex individually. The MDA-Lite, however, reserves node control for particular cases and proceeds hop by hop in the general case. At each hop it seeks to discover all of the vertices at that hop, and in doing so discovers some portion of the edges between that hop and the prior hop. It then seeks out the remaining edges. It operates on the assumption that the diamonds that it encounters will be uniform and unmeshed. If

this assumption holds, hop-by-hop probing will maintain the MDA's failure probability bounds. Because these two topology assumptions might not hold, the MDA-Lite tests for a lack of uniformity and the presence of meshing using methods that are less costly than full application of the MDA. When it detects a diamond that does not adhere to one of the assumptions, it switches to the MDA. These steps are described below.

Uniform, unmeshed diamonds

The MDA-Lite, operating on the assumption that a hop is uniform, sends probes to that hop without node control. It starts by reusing one flow identifier from each of the vertices that it has discovered at the previous hop, continuing with additional previously-used flow identifiers and then new ones. It applies the MDA's stopping rule to remain within the MDA's failure probability bounds for vertex detection.

To take as examples the topologies in Fig. 3.1, the MDA-Lite sends n_4 probes to hop 2, n_2 probes to hop 3, and n_1 probes to hop 4. Discovery of all vertices in the diamond therefore requires $n_4 + n_2 + 2n_1$ probes, or 68 probes when applying the values in Veitch et al.'s Table 1, regardless of whether the diamond is unmeshed or meshed. This compares to the numbers for the MDA that we determined above: $99 + \delta$ probes for the unmeshed diamond and $163 + \delta'$ probes for the meshed diamond.

Discovering all of the vertices at adjacent hops i and $i + 1$ does not imply that the MDA-Lite will have discovered all of the edges. Finishing up the edge discovery is straightforward, though, for unmeshed hops, in the sense that it is deterministic rather than stochastic. It consists of tracing backward from each vertex at hop $i + 1$ that does not yet have an identified predecessor or forward from each vertex at hop i that does not yet have an identified successor. There are three cases to consider:

- Hop $i + 1$ has fewer vertices than hop i. For each hop i vertex that does not yet have an identified successor, the flow identifier of a probe that has discovered that vertex is used to send a probe to hop $i + 1$. Assuming no meshing, this completes the edge discovery.
- Hop $i + 1$ has more vertices than hop i. For each vertex at hop $i + 1$ that does not yet have an identified predecessor, the flow identifier of a probe that has discovered that vertex is used to send a probe to hop i. Assuming no meshing, this completes the edge discovery.
- Hop $i + 1$ has the same number of vertices as hop i. We apply both of the methods just explained above.

Because a diamond could be meshed or non-uniform, the MDA-Lite tries to detect those cases, as described below.

Detecting meshing

To detect meshing, stochastic probing is required, and this involves a limited application of node control. For a pair of hops having two or more vertices each, the meshing test consists of tracing from the hop with the greater number of vertices to the one with the lesser number of vertices, or tracing in either direction if the hops have equal numbers of vertices. When tracing forwards, meshing is detected if any predecessor vertex has an

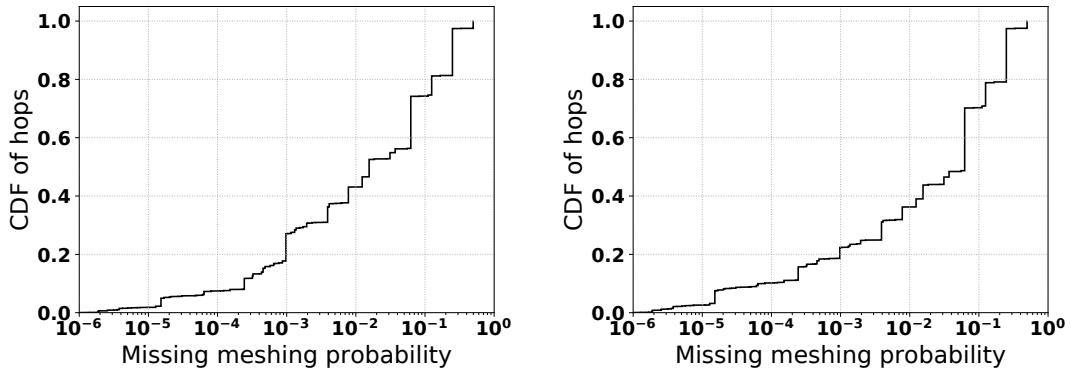
out-degree of 2 or more. For backwards tracing, it is if any successor vertex has an in-degree of 2 or more. The test requires node control: We introduce a parameter, $\phi \geqslant 2$, for the MDA-Lite, which determines the number of flow identifiers that have to be generated for each vertex at the hop from which tracing will begin. Probes with these flow IDs are sent to the other hop.

The probability of failing to detect meshing depends upon ϕ. We calculate this probability as follows. Suppose that the test is through forward tracing, and let V be the set of two or more vertices at hop i and let $\sigma(v)$ designate the set of successor nodes of a vertex $v \in V$. When ϕ flow IDs are generated for each vertex $v \in V$ and probes with those flow identifiers are sent to hop $i + 1$, the probability of failing to detect meshing is:

$$\prod_{v \in V} \frac{1}{|\sigma(v)|^{\phi-1}} \tag{3.1}$$

This probability calculation extends with trivial adjustments to the case of backward tracing.

A minimum value $\phi = 2$ is required in order to detect meshing. Whether to use a higher value, with a lower failure probability, is up to the MDA-Lite implementation. We examined how well this minimum value would work on the meshed diamonds identified by the MDA in the survey that is described in Sec. 3.5.1. Looking at the topology of each hop pair, we calculated the probability of the MDA-Lite failing to detect the meshing. We did this both for *measured diamonds*, which is to say that each diamond is weighted by the number of times that it is encountered in the survey, and for *distinct diamonds*, in which we weight each diamond just once, regardless of how many times it has been seen. Fig. 3.2 plots CDFs for the probability of the MDA-Lite with $\phi = 2$ missing meshing at a hop pair for which the MDA detected meshing. We see that, for both measured and



(a) Measured (b) Distinct

Figure 3.2: The probability of failing to detect meshing

distinct diamonds, the probability of failing to detect meshing is 0.1 or less on 70% of meshed hop pairs and 0.25 or less on 95% of the cases. If we consider this to be too high a probability, ϕ is tunable, and we can set it to 3 or 4.

The overhead generated by the meshing test is lower than the overhead of the MDA's use of node control. Even with a value of ϕ of 3 or 4, this is lower than $n_1 = 9$, the minimum number of flow identifiers per vertex required by the MDA's use of node control

in Veitch et al.. Furthermore, the MDA-Lite's meshing test is applied only to a minority of diamonds. As previous surveys have shown, and our survey confirms, nearly half of all diamonds consist of only a single multi-vertex hop (48% for measured and 45% for distinct diamonds). The MDA-Lite's meshing test only applies where there are two adjacent multi-vertex hops, but the MDA applies node control whenever there is a multi-vertex hop.

Detecting non-uniformity

Once edge discovery is complete, and if the MDA has not been engaged because of meshing, the MDA-Lite tests for non-uniformity. The test is a purely topological one because the MDA-Lite makes the same assumption as the MDA about the evenness of load balancing: that each load balancer dispatches flow IDs in a uniform manner. (Based upon our experience, this appears to be a realistic assumption, but a survey on this particular point would be worthwhile.) What we term "width asymmetry" in our survey (see Sec. 3.5) is therefore the indicator of non-uniformity.

The MDA-Lite detects width asymmetry as follows. For a pair of hops i and $i+1$, if the number of successors is not identical for every vertex at hop i or if the number of predecessors is not identical for every vertex at hop $i+1$, the diamond has width asymmetry and is considered to be non-uniform, and the MDA-Lite switches over to the MDA.

Finding non-uniformity depends upon the topology in question having been fully revealed. Unlike the MDA, the MDA-Lite does not provide statistical guarantees on full topology discovery. Rather, the MDA-Lite assumes that any non-uniformity is likely to be low so that the full topology will most probably be revealed. We empirically justify this assumption based upon our survey results in Sec. 3.5.1.

3.2.4 MDA-Lite evaluation

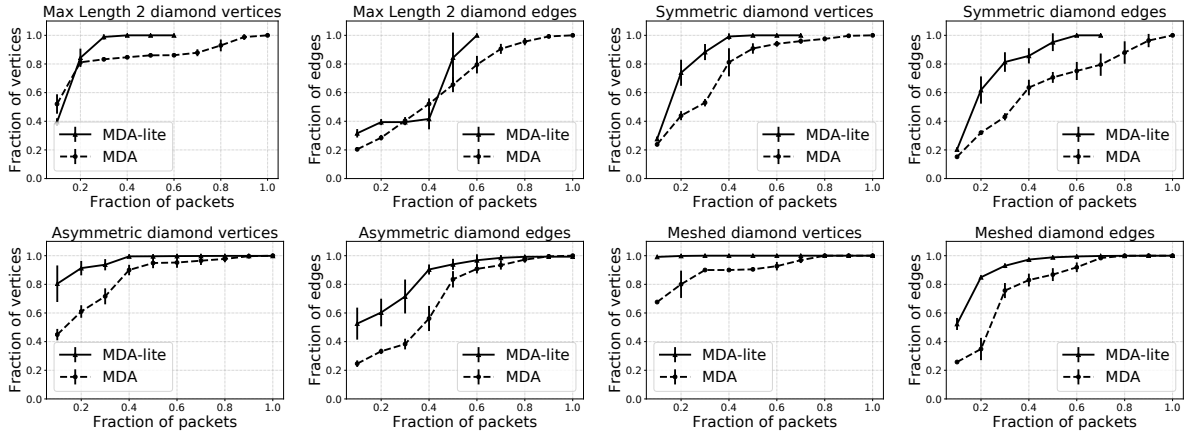


Figure 3.3: MDA-Lite versus MDA simulations

We have tested the MDA-Lite both through simulations and measurements on the Internet, finding in both cases that it compares favorably to the full MDA.

Evaluation through simulations

Simulations allow us to compare the MDA-Lite to the MDA on known topologies and in an environment free of factors, such as variations in router load, that are not related to the algorithms. We have chosen topologies based on both the categories of diamond that are relevant to the MDA-Lite (uniform or asymmetric, meshed or not, see Sec. 3.2.3), and on what we found in our survey (Sec. 3.5.1).

- The *max length 2 diamond*, found on the trace `pl2.prakinf.tu-ilmenau.de` to `83.167.65.184`, consists of a divergence point, a 28 vertex hop, and a convergence point. Nearly half of all diamonds in the survey are of maximum length 2, though this is a particularly wide example. Where the MDA will perform node control on each of the 28 vertices, the MDA-Lite will avoid doing so. Finding no adjacent multi-vertex hops, the MDA-Lite will not apply its meshing test.

- The *symmetric diamond*, found on the trace `ple1.cesnet.cz` to `203.195.189.3`, has three multi-vertex hops, with 10 being the most vertices at a hop. There is no meshing between the hops. On this diamond, the MDA-Lite will be obliged to perform a light version of node control in order to test for meshing. Finding none, it will not switch over to the full MDA.

- The *asymmetric diamond*, found on the trace `kulcha.mimuw.edu.pl` to `61.6.250.1`, has nine multi-vertex hops, with 19 being the most vertices at a hop. The edges are laid out in such a way that at least one of the hops is not uniform, which is to say that there is a greater probability of a probe packet with an arbitrarily chosen flow identifier reaching some vertices at that hop rather than others. It has a "width asymmetry" of 17 (this metric is defined in Sec. 3.5). It is unmeshed. If the MDA-Lite discovers the asymmetry, it will be obliged to switch over to the full MDA.

- The *meshed diamond*, found on the trace `ple2.planetlab.eu` to `125.155.82.17`, has five multi-vertex hops, with 48 being the most vertices at a hop. It is meshed, and if the MDA-Lite discovers the meshing it will be obliged to switch over to the full MDA.

The simulations ran on Fakeroute, the tool that we describe in Sec. 3.4. Fig. 3.3 shows the results of 30 runs on each of the four topologies, with vertex discovery graphs on the left and edge discovery graphs on the right. Two curves are plotted on each graph: one for the MDA-Lite with $\phi = 2$ and one for the MDA. The portion of the topologies' vertices or edges discovered as each algorithm is running is plotted on the vertical axis. The horizontal axis indicates the number of probe packets sent, normalized to 1.0 being the number of packets sent by the MDA in a given run. Since the MDA-Lite sends fewer packets when confronted with max length 2 and symmetric diamonds, its curves stop before reaching the right hand side of the graph. Error bars with standard deviation are given. We see that the MDA-Lite tends to discover more of these topologies faster than the MDA, though not always, and that it discovers the entire topology sooner. In cases where it does not need to switch over to the full MDA, it also economizes on the number of probes that it sends, reducing by 40% the full MDA's overhead on these examples. For these cases, we see that the MDA-Lite is not sacrificing the ability to discover the full topology. Because it is more economical in its use of probes, it discovers more faster. When it does not have to switch over to the full MDA, it uses significantly fewer probes.

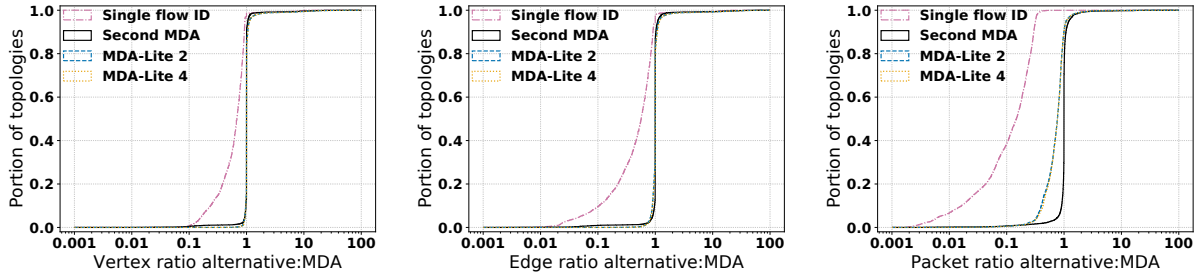


Figure 3.4: Comparative performance: CDFs over 10,000 measurements in the Internet

In the other cases, although it discovers the full topology faster than the MDA, the switch to the full MDA means no economy in its use of probes.

Evaluation through measurements

We performed our measurement-based evaluation on a sample of 10,000 source-destination pairs from our survey (Sec. 3.5.1) for which diamonds had been discovered. For each of these, we ran five variants of Paris Traceroute successively: two with the MDA; one with the MDA-Lite and $\phi = 2$; one with the MDA-Lite and $\phi = 4$; and one with just a single flow I. As a reminder, the parameter ϕ, defined in Sec. 3.2.3, governs how much effort the MDA-Lite will expend in trying to detect meshing.

For each topology, the first run with the MDA serves as the basis for comparing the other algorithms. We calculate the ratio of vertices discovered, edges discovered, and packets sent. The results, plotted as CDFs, are shown in Fig. 3.4. The horizontal axis plots the ratios in log scale, with 10^0 indicating that the algorithms performed the same. For the vertex and edge discovery plots, a value to the left of this value indicates that the competing algorithm discovered less than the first MDA run, and so performed worse, and a value to the right indicates that it discovered more, and so performed better. For the packets plot, at 1, the tools sent the same number of packets, whereas a value to the left of this indicates that the competing algorithm sent fewer packets than the first MDA run, and so performed better, whereas a value to the right of this indicates that the competing algorithm sent more packets than the first MDA run, and so performed worse.

We run the MDA algorithm twice because there are variations from run to run, both because of changing network conditions and because of the stochastic nature of MDA and MDA-Lite discovery. The second MDA will sometimes perform better, sometimes worse than the first, and its curve, shown as a solid black line in the plots, forms the basis against which to compare the other algorithms. While the second MDA performs close to the first, it discovers fewer vertices 12% of the time and more vertices 12% of the time; fewer edges 20% of the time and more edges 20% of the time. We believe that these differences are largely attributable to the stochastic nature of the MDA, meaning that either the first or the second run occasionally terminates its discovery process without having discovered all of the vertices (and hence edges) that are available to discover. Recall from Sec. 3.2 that the MDA's failure bound for discovering the successors to a vertex is set as a function of a global failure bound for the entire topology and a maximum number of branching points that the topology might have. This latter parameter is set to 30 by default, but in

	Vertices	Edges	Packets
MDA 2	0.998	0.999	1.005
MDA-Lite $\phi = 2$	1.002	1.007	0.696
MDA-Lite $\phi = 4$	1.004	1.005	0.711
Single flow ID	0.537	0.201	0.040

Table 3.1: Comparative performance on aggregated topology: ratios with respect to a first MDA round over 10,000 measurements in the Internet

complex topologies of the sort that we have encountered in our survey, there can be far more branching points.

For the comparison between the MDA and the MDA-Lite, we observe that there is no discernible difference between $\phi = 2$ and $\phi = 4$ for the MDA-Lite. Most importantly, the MDA-Lite performs nearly identically to the second MDA run with respect to the first MDA run: sometimes better, sometimes worse. Compared to the first MDA run, the MDA-Lite performed better 14% of the time and worse 14% of the time for the vertices; better 20% of the time and worse 26% of the time for the edges. We attribute the larger number of instances of worse performance to the occasional failure of MDA-Lite to detect meshing or non-uniformity. The impact of this greater number on overall performance is negligible, as the ratio curves are hard to distinguish.

Paris Traceroute with a single flow ID performs notably worse on the whole than the MDA in both vertex and edge discovery. In only 12% of the cases, we observed at least 90% of the vertices and in only 10% of the cases, we observed at least 90% of the edges. We did detect some outliers where Paris Traceroute with a single flow ID discovers a greater number of vertices and edges than the MDA. These correspond, we believe, to cases where the route changed between the runs.

The other aspect of performance that concerns us is the number of packets that were sent. In 89% of the comparisons, the MDA-Lite realized probe savings. We find that we save 40% of the probes on 30% of the topologies. The ratio curves for both $\phi = 2$ and $\phi = 4$ are nearly identical and they are clearly superior to the curve for the second MDA run, meaning that when there is a diamond in the topology, the MDA-Lite will tend to use significantly fewer packets that the MDA.

Paris Traceroute with a single flow ID sends many fewer packets. The cost of discovering an entire multipath topology via the MDA can be anywhere from less than 2 times more to 1000 times more than the cost of tracing a single route with a single flow ID.

From a macroscopic point of view, Table 3.1 provides results on the overall topology formed by the aggregation of the 10,000 measurements of the evaluation dataset. Ratios of topology discovered and probes sent are computed with respect to the first MDA. We see that the topologies discovered by the MDA and the MDA-Lite are very close, with a maximum of 0.7% difference for the edges. We see also that the MDA-Lite cuts the number of probe packets sent by roughly 30%. Paris Traceroute with a single flow ID sends only 4% of the packets sent by the first MDA, but only discovers 53.7% of the vertices and 20.1% of the edges.

3.3 Multilevel route tracing

The third principal contribution of this paper, after the MDA-Lite and Fakeroute of the previous sections, is IPv4 multilevel route tracing, embodied in a version of Paris Traceroute that we refer to here as Multilevel MDA-Lite Paris Traceroute (MMLPT). By "multilevel", we mean that the tool provides router-level information in addition to the standard interface-level information. Some router-level information is already commonly provided by standard Traceroute command line tools, as they perform DNS look-ups on the IP addresses that they discover, and the name of an interface is often a variant on the name that has been assigned to the router as a whole. In addition, some of the prior work [125, 43] that we describe in Sec. 3.6 can reveal router or middlebox level information in the context of a Traceroute. Within the network measurement community, there are survey workflows, such as the one employed by `bdrmap` [95], that perform route traces and then alias resolution, and there are survey tools, such as `scamper` [91], that are capable of performing both functions independently. To take another recent example, Marchetta et al. [100], employed a specialized tool, Paris Traceroute with the MDA, to conduct multipath tracing, and then another specialized tool, MIDAR, to conduct alias resolution on the IP addresses that the first tool reveals. But there has not previously been a command-line Traceroute tool, in the line of Van Jacobson's Traceroute [78], Modern Traceroute for Linux [25], and the like, with an option to obtain a router level view of multipath routes. With the advent of multipath route tracing ten years ago, it would seem to be a natural next step to incorporate alias resolution directly into Traceroute itself. Such a tool could readily be slotted in to workflows that currently invoke a Traceroute, and it would bring new capabilities to those, such as network operators, who use Traceroute for network troubleshooting purposes.

Alias resolution from a Traceroute perspective, coming as it does from a single route trace from a single vantage point, will never be as complete as alias resolution performed from multiple vantage points on IP addresses gleaned from traces from multiple vantage points. Nevertheless, we argue, alias resolution integrated into Traceroute, provides valuable information. When one observes multiple parallel paths in a route trace, the question immediately arises as to whether they are independent or not. Between two adjacent hops, one could be observing links to different interfaces on a single router or links to separate routers. MMLPT provides the capacity to distinguish between these cases at the moment of the route trace, without having to apply an additional tool for post hoc analysis, such Marchetta et al in [100]. Anyone who conducts route traces outside of the context of a dedicated survey, such as a network operator performing troubleshooting, can benefit.

The remainder of this section describes the alias resolution techniques that MMLPT employs (Sec. 3.3.1) and shows how we evaluate them (Sec. 3.3.2). Survey results using the tool are reported in Sec. 3.5.2.

3.3.1 Alias resolution

MMLPT performs alias resolution using MIDAR's Monotonic Bounds Test (MBT) [87] and two techniques described by Vanaubel et al.: Network Fingerprinting [125], and MPLS Labeling [123]. In its overall approach, it follows the MBT's set-based schema for alias identification. An initial set is established of all of the candidate addresses, and then broken down into smaller and smaller sets as probing evidence indicates that certain pairs

of addresses are not related. The sets are composed in such a way that each address in a set has failed alias tests with every address in every other set. At any point, each set that contains two or more addresses is considered to consist of the aliases of a common router. Further probing further refines these sets.

MIDAR faces a particular challenge in establishing its initial sets of candidate aliases, as it is designed to seek aliases from on the order of a million candidate addresses. It breaks this large number down into manageable sized initial sets by sorting aliases on the basis of how fast their IP IDs are evolving over time. MMLPT skips this step, as its task is narrower: to seek aliases among the addresses found in a single multipath route trace. It assumes that the aliases of a given router are to be found among the addresses found at a given hop, and so there will be at most on the order of one hundred candidate aliases. As a result, we only borrow the MBT from MIDAR, and not its full complement of probing stages and heuristics.

Evidence that two addresses are not related comes in different forms, depending upon the test:

- The MBT looks at sequences of IP IDs from addresses that have been probed alternately. A monotonic increase in identifiers, taking wraparound into account, is consistent with the addresses being aliases, whereas a single out-of-sequence identifier is used to place the addresses into separate alias sets. We recall that MMLPT has used UDP indirect probing and that we have used MIDAR with UDP, TCP, and ICMP direct probing to collect IP ID time series.

- Network Fingerprinting looks at the TTLs of reply packets to a ping style probe and a Traceroute style probe, and infers their likely initial TTLs. Replies to probes of different addresses having different initial TTLs are almost certainly from different routers, and so the addresses are placed into separate alias sets.

- MPLS Labeling looks at the MPLS labels that appear in reply packets from different addresses. Vanaubel et al. [123] have characterized the different cases of MPLS tunnels with load balancing and developed methods to infer aliases from MPLS labels. To be usable, labels of interfaces in an MPLS tunnel have to be constant over time for each interface. Otherwise, MPLS labels are not helpful to infer aliases. Then, if, for two interfaces in an MPLS tunnel found at the same hop, their labels differ, it is highly likely that these two interfaces belong to two different routers. So the addresses are placed into separate alias sets. Conversely, if the labels are the same for the two interfaces, then it is highly likely that these two interfaces belong to the same router.

False positives, in which two addresses that are not aliases remain in the same set, can arise through their routers having identical fingerprints and MPLS signatures (when available), alongside a lack of sufficient MBT probing. False negatives, in which two addresses that are in fact aliases get placed in separate sets can arise when, instead of a single router-wide IP ID counter, a router employs separate IP ID counters for each flow identifier, and so the addresses fail the MBT [53].

Some of the basic data required by these techniques is collected as part of basic MDA-Lite Paris Traceroute probing: IP IDs that are used by the MBT; the TTLs of "indirect probing" reply packets that are used by Network Fingerprinting; and the MPLS labels

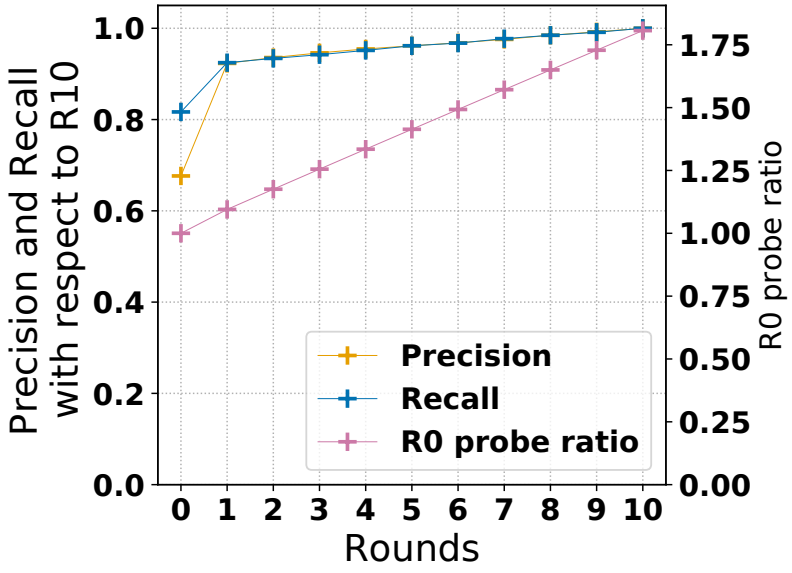


Figure 3.5: Alias resolution over ten rounds

that appear in reply packets. A light version of the MBT, along with MPLS Labeling, can therefore be performed "for free", based on these data. The results are then refined by MMLPT over additional rounds of probing, with the direct probes required for Network Fingerprinting and indirect probes to solicit more and longer sequences of IP IDs for the MBT. The signature-based methods are applied just once, whereas successive rounds of the MBT refine the results. After 10 rounds, MMLPT declares sets that remain as aliases.

Our tool, like MIDAR, produces three possible outcomes for a pair of IP addresses. Either it accepts that they are aliases of the same router, or they are rejected as being aliases of the same router, or it is not possible for the tool to determine one way or the other. Failure to determine is not an unusual case, as there are addresses from which responses to probes do not have monotonically increasing IP ID values. Such an address might, for instance, systematically respond with the same value in response to every probe. Or it might not provide a sufficient number of responses from which to construct a time series.

3.3.2 Evaluation

We looked at how MMLPT's alias resolution results evolve round by round. Round 0 is based on just the data obtained through MDA-Lite Paris Traceroute, with no additional probing. The MBT and signature-based tests are applied to the extent possible. Round 1 adds one direct probe to each of the IP addresses at a given hop, in order to provide more complete Network Fingerprinting signatures. It also is the first round of MBT probing, attempting to elicit 30 replies per address. Each subsequent round through to Round 10 consists of an additional 30 indirect probes per address, in order to further refine the alias sets using the MBT.

Fig. 3.5 presents overall values for precision, recall, and numbers of probes sent over the 10,000 measurements conducted for the MDA-Lite evaluation of Sec. 3.2.4. We do not

	Accept Direct	Reject Direct	Unable Direct
Accept Indirect	0.365	0.005	0.283
Reject Indirect	0.144	N/A	N/A
Unable Indirect	0.203	N/A	N/A

Table 3.2: Findings for 4798 address sets identified as routers either by indirect probing (MMLPT) or direct probing (MIDAR), expressed as portions adding up to 1.0

have ground truth, so precision and recall are relative to our best available determination of the alias sets, which is the result of Round 10 in each case. The number of probes is relative to the number sent in Round 0.

Round 0, with no probing beyond that which is performed for MDA-Lite Paris Traceroute, yielded 68% precision and 81% recall with respect to the Round 10 results. A significant jump to 92% in both cases came with a first round of probing, and then there was a slow increase with each successive round. The additional probing for each round was less than 10% of the basic MDA-Lite Paris Traceroute probing.

These results indicate that we can glean router-level information with a modest amount of additional probing, typically 20% more is enough to get a precision and a recall greater than 92% with respect to round 10, and 75% more to complete the ten rounds. Additional work will be required in order to better establish a firm basis against which to compare, so as to provide clearer guidance on the tradeoff between probing and the completeness and accuracy of the results.

We also looked at the potential benefits and costs of adding direct probing, as we had implemented MMLPT with only indirect probing, for the MBT. For each diamond, MMLPT identifies zero or more address sets as routers, validating or rejecting address sets via indirect probing. We compare these results with what direct probing IP ID techniques would have found, using MIDAR for this. We ran MIDAR on all the addresses of the diamond, and MIDAR too identifies zero or more address sets as routers in the diamond. We take the union of the address sets identified by both tools, and compare: which ones did both accept as being a router, and which ones were accepted by one of the tools but not by the other? If a tool does not accept an address set, it is either because it has rejected it (for instance by finding a pair of addresses that has failed the MBT) or because it is unable to determine if one or more of the addresses belongs in the set (for instance because of an insufficient time series from an address).

Table 3.2 shows the results for 4798 address sets, of which 3414 were identified as routers by MIDAR and 3140 by MMLPT. The values are the portion of address sets that fall within each category. 36.5% were accepted as routers by both MIDAR and MMLPT. Just 0.5% of sets accepted by MMLPT are rejected by MIDAR, whereas 14.4% of sets accepted by MIDAR are rejected by MMLPT. The latter can be explained by routers that implement per-interface counters for the IP ID for the ICMP Time Exceeded messages associated with indirect probing and router-wide counters for the ICMP Echo Reply messages associated with direct probing.

Significant portions of sets accepted by one tool encounter a failure to determine a result by the other tool: 20.3% of sets accepted by MIDAR led to no conclusion by

MMLPT and 28.3% of sets accepted by MMLPT led to no conclusion by MIDAR. Upon further investigation, we found that 98.6% of the non conclusive cases for MMLPT are due to either constant (mostly zero) IP IDs and 1.4% to non monotonic IP ID series. Looking at MIDAR logs, we found that the 28.3% inconclusive cases had different causes: for each inconclusive set, at least one IP in the set was either unresponsive to direct probing (60.5%), or its IP ID series was a copy of the probe IP ID (22.8%), or its IP ID series was non monotonic (13.6%), or MIDAR got unexpected responses, meaning that the reply did not match that which would be expected based upon the probe protocol used (3.1%).

Our overall conclusion is that direct probing provides a potentially valuable complement to indirect probing, and that we should include it in future versions of MMLPT, while also evaluating the tradeoff in what is gained against the additional probing cost that it will entail.

3.4 Fakeroute

For any given multipath route between source and destination, one can calculate the precise probability of the MDA failing to detect the entire topology. This calculation is a simple application of the MDA's stopping rule with the chosen stopping points, the values n_k described in Sec. 3.2.1, along with the basic assumptions underlying the MDA, such as that load balancing will be uniform at random across successor vertices [126, Sec. II.A]. For a vertex in the topology that has $K > 1$ successors, the first successor will certainly be found by the first probe packet (among the assumptions is that all probes receive replies), but there is a probability $1/K^{n_1-1}$ that a total of n_1 probes will fail to discover a second successor, and the probabilities of failing to discover each of the remaining successors $k \leqslant K$ are similarly straightforward to calculate. Veitch et al. provide the details [126, Sec. II.B].

In principle, therefore, it should be possible to test that the MDA has been correctly implemented by a software tool by running it repeatedly on a suite of benchmark topologies and seeing that the failure probabilities are as predicted. For scientific purposes, we would want, if at all possible, to verify the conformance of a tool before using it, but we have not had that capability until now for tools that implement the MDA. Our contribution is a network multipath topology simulator that takes as input a given topology and a number of values n_k that is at least equal to the highest branching factor encountered in the topology, that calculates the probability that the MDA will fail to discover the full topology, and that runs the actual software tool in question repeatedly on the topology to verify that the tool does indeed fail at the predicted rate, not more, not less, providing a confidence interval for this result.

Our Fakeroute is a complete rewrite of the Fakeroute tool that has been provided as part of the *libparistraceroute* library [14], and which enabled small numbers of runs of a tool on a simulated topology, for simple debugging purposes, but that was not designed for large numbers of runs with statistical validation. The new Fakeroute, written in C++, uses *libnetfilter-queue* [13] to sniff probe packets sent by a tool and suck them into the simulated environment rather than letting them out of the host into Internet. Once a probe is in, Fakeroute uses *libtins* [15] to read the flow identifier and TTL from its header fields. These are used to simulate the probe's passage through the topology, with the

pseudo randomness of load balancing being emulated by the Mersenne Twister [2] that comes with the standard C++ library. Using *libtins*, Fakeroute crafts either an ICMP Time Exceeded or an ICMP Port Unreachable reply depending on whether the probe is determined to have reached an intermediate router or the destination, and sends that back to the tool. For example, on a topology with the simplest possible diamond (a divergence point, two nodes, and a convergence point), we were able to test that the real failure probability of the topology, which is 0.03125, given the set of n_k values used by the MDA for a failure probability of 0.05, was respected. We ran the MDA 1000 times on this topology to obtain a sample mean rate of failure, and obtained 50 such samples to obtain an overall mean and a confidence interval. This took 10 minutes on a contemporary laptop machine, giving a 0.03206 mean of failure, with a 95% confidence interval of size 0.00156. We were able to run the same test on much larger topologies as well, as indicated in the previous section. Fakeroute is available as free open-source software at the URL mentioned at the end of Sec. 4.1.

3.5 Surveys

This section presents the two surveys that we have conducted, one at the IP level, the other at the router level. The aim in both is to characterize the topologies that are encountered by multipath route tracing in the IPv4 Internet, along the lines of earlier surveys [42, 100, 35] mentioned in the Related Work section.

Our focus is on the "diamonds" (see Sec. 3.2.1 for the definition) that are encountered in a route trace. We define a a **distinct diamond** by its divergence point and its convergence point. This means that if a diamond is encountered multiple times in the course of a survey, there might be differences in its measured internal topology from one encounter to the next. If either a divergence point or a convergence point is non-responsive (a "star" in common parlance), we consider it as different from a diamond that has responsive divergence and convergence points, even if the two diamonds have other IP addresses in common. Since a diamond might show up in multiple measurements, we define each encounter with a distinct diamond to be a **measured diamond**. Each way of counting reflects a different view of what is important to consider: the number of such topologies, or the likelihood of encountering one. We look at both.

The surveys describe how large diamonds are, both in number of hops and in number of vertices at a given hop. Also, because we have found that "uniformity" and "meshing" are relevant to the ability to economize on probes when tracing at the IP level (see Sec. 3.2.2), we describe these features. For the metric definitions that follow, we apply those of Augustin et al. [42] for "maximum width" and "maximum length" and add "maximum width asymmetry" and "ratio of meshed hops". As illustrated in Fig. 3.6, these are:

The **maximum width** is the maximum number of vertices that can be found at a single hop, as in the boxed hop of the left-hand diamond.

The **maximum length** is the length of the longest path between the divergence and the convergence point, as shown by the set of bold edges in the left-hand diamond.

The **maximum width asymmetry** is a topological indicator of a lack of uniformity. We define it first for a pair of hops i and $i+1$.

- If hop i has fewer vertices than hop $i+1$, it is the maximum difference in the number of successors between two vertices at hop i.

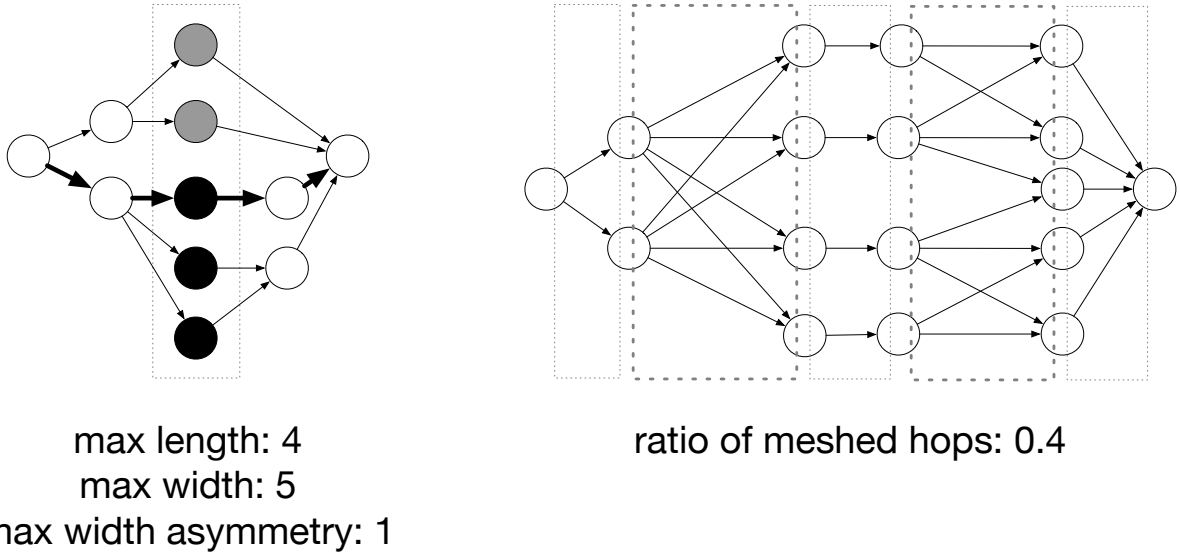


Figure 3.6: Diamond metrics

- If hop i has more vertices than hop $i+1$, it is the maximum difference in the number of predecessors between two vertices at hop $i+1$.
- If hops i and $i+1$ have identical numbers of vertices, it is the maximum of the two values described above.

For a diamond as a whole, it is the largest value of maximum width asymmetry found across all hop pairs, as shown by the grey and black vertices of the left-hand diamond.

The **ratio of meshed hops** of a diamond is the portion of hop pairs of hops that are meshed, as shown in the right-hand diamond, in which two of the five hop pairs are meshed, for a ratio of 0.4.

3.5.1 IP level survey

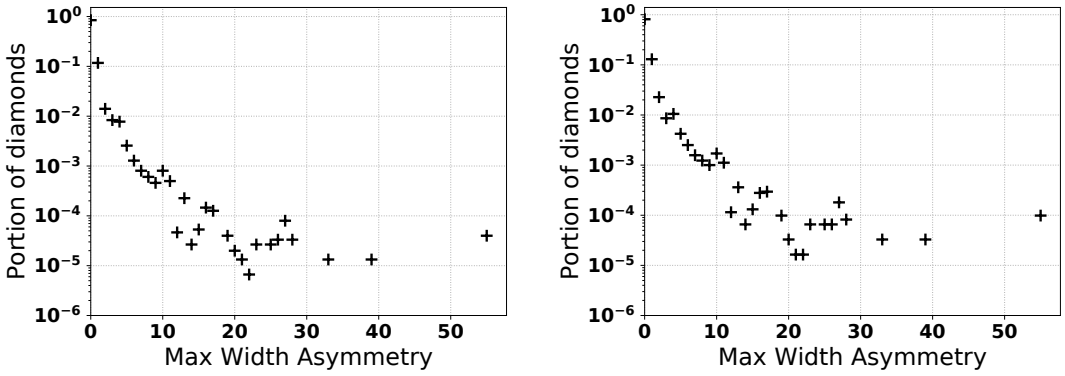
The IP level survey is based on multipath route traces from 35 sources towards 350,000 destinations during two weeks starting 8 March 2018.

The route tracing tool was the *libparistraceroute*-based MDA Paris Traceroute [14], using its default parameters. We employed UDP probes, as Luckie et al. [98] found best results for discovering load balanced paths with such probes.

The sources were PlanetLab nodes running Fedora 24 or 25, obtained through PlanetLab Europe [19]. (We also ran a survey with similar results, which can be found at the URL mentioned at the end of Sec. 4.1, on the new EdgeNet infrastructure [8] affiliated with PlanetLab Europe.)

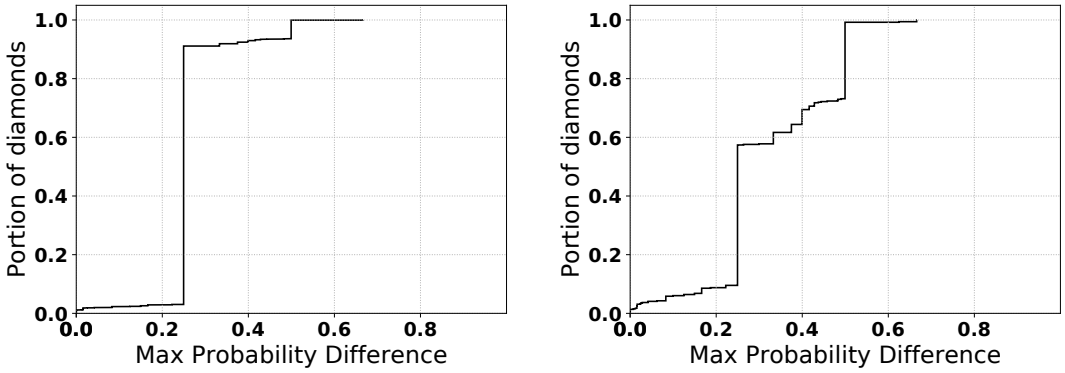
The destinations were chosen at random from the IPv4 addresses rated as "highly responsive" in the Internet Address Hitlist IMPACT dataset `Internet_address_hitlist_-it78w-20171113`, ID DS-822, covering 17 January 2015 to 15 December 2017 [30].

We discarded route traces that we could not collect because of infrastructure troubles, yielding 294,832 exploitable results, among which 155,030 passed through at least one per-flow load balancer. There were 60,921 distinct and 220,193 measured diamonds.



(a) Measured (b) Distinct

Figure 3.7: Width asymmetry



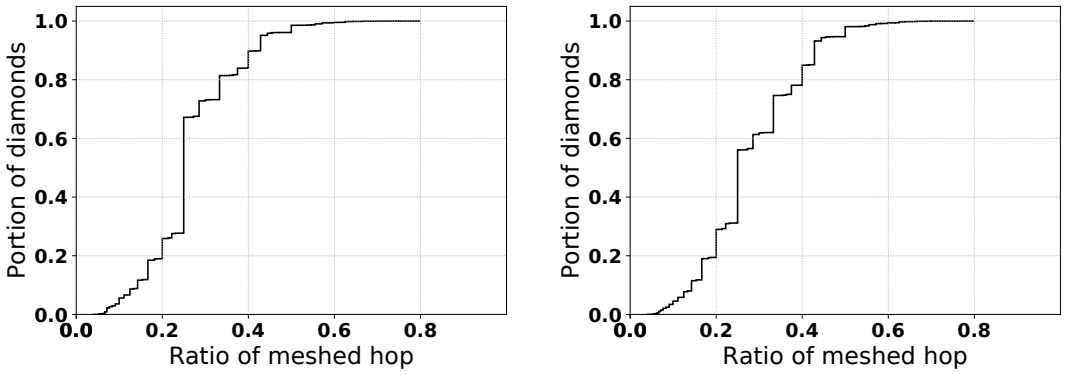
(a) Measured (b) Distinct

Figure 3.8: Maximum probability difference in width-asymmetric diamonds

We start by looking at uniformity and meshing.

Uniformity In both measured and distinct diamond asymmetry distributions (Fig. 3.7), 89% of diamonds have zero asymmetry. This means that most diamonds are uniform, provided that load balancing is uniform across next hop interfaces, and supports the MDA-Lite's assumption of uniformity. But if the MDA-Lite cannot detect the asymmetry in a diamond that is among the 11% that are asymmetric, it will not switch over to the full MDA and it risks failing to discover the full topology. It is most likely to encounter difficulty on an unmeshed diamond, as, when meshing is detected, the full MDA is invoked. Only 2.3% of measured and 3.6% of distinct diamonds are both asymmetric and unmeshed. We examined these diamonds for differences in discovery probability among vertices at a common hop, plotting the CDFs of all non-zero probability differences in Fig. 3.8. In these cases, 90% of measured and 58% of distinct diamonds have a maximum probability difference of 0.25 and, for both, 99% have a maximum probability difference of 0.5. This indicates that the MDA-Lite is very unlikely to fail in uncovering a lack of uniformity, which is borne out by our experimental results in Sec. 3.2.4. This issue could be more

rigorously studied with further mathematical analysis.



(a) Measured (b) Distinct

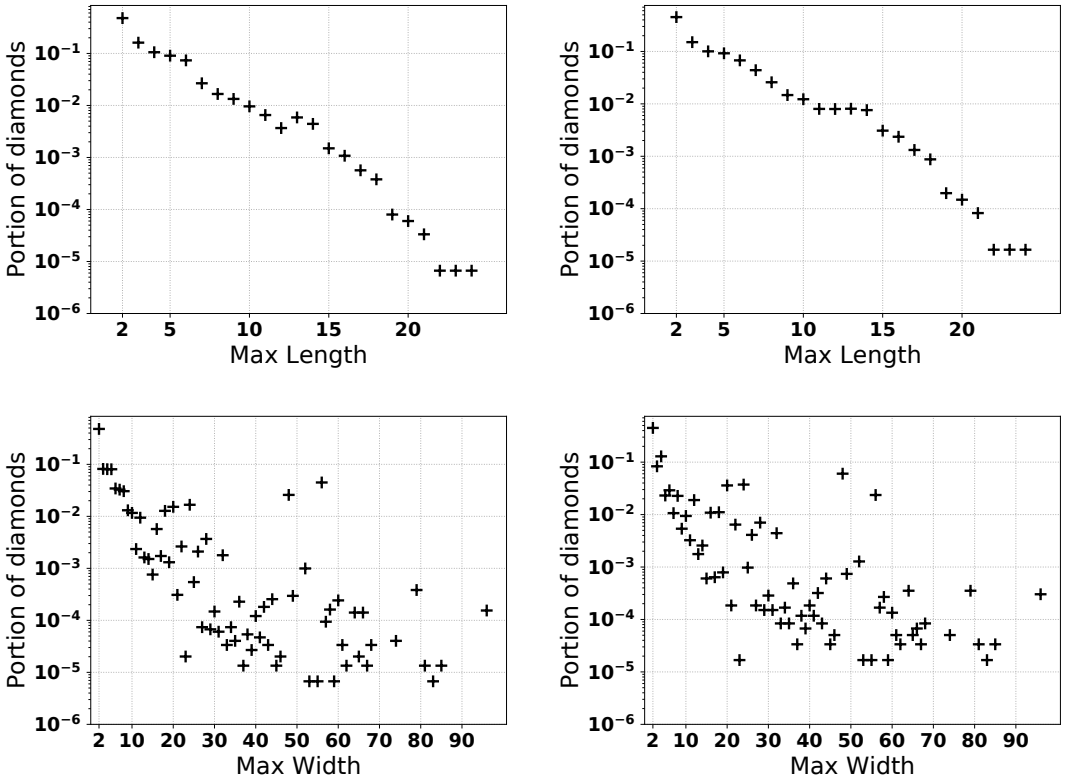
Figure 3.9: Ratio of meshed hops

Meshing Of the 220,193 measured diamonds in our survey, 32,430 present at least one meshed hop, and of the 60,921 distinct diamonds, 19,138 are meshed. Fig. 3.9 plots CDFs of the ratio of meshed hops for the meshed diamonds. The MDA-Lite offers probe savings over the full MDA when a pair of hops is not meshed. More than 80% of meshed diamonds have a ratio of of meshed hops under 0.4, which indicates a significant potential for the MDA-Lite to realize significant probe savings, even on meshed diamonds. We continue by looking at the length and width metrics, for which the distributions are shown in Fig. 3.10. Almost half of both measured and distinct diamonds have a maximum length of 2, meaning that they consist of a divergence point, a single multi-vertex hop, and a convergence point. The MDA-Lite is more economical than the full MDA on such diamonds. The largest value of maximum width encountered is 96. Such a high value is unprecedented, with earlier surveys [42, 100] reporting maximum widths of at most 16. A notable feature of the maximum width distributions is their peaks at 48 and 56. Further investigation indicates that the distinct diamond distribution might be overstating what is in fact being encountered by the route traces. Though the diamonds are distinct by our definition, meaning that they have a unique pair of divergence and convergence points, they share a large portion of their IP addresses. This suggests a common structure that is being frequently encountered via a variety of ingress points.

Looking at the joint distributions of maximum width and maximum length (Fig. 3.11), we see that short and narrow diamonds continue to be the most common, as found in previous surveys. For example, we found that 24.2% of measured and 27.4% of distinct diamonds were of maximum length 2 and maximum width 2, corresponding to the simplest possible diamond. The maximum width 48 and 56 diamonds also reveal themselves to have a variety of different maximum lengths.

3.5.2 Router level survey

The router level survey is based upon the 155,030 route traces from the IP level survey that passed through at least one load balancer. We retraced these with Mutilevel MDA-



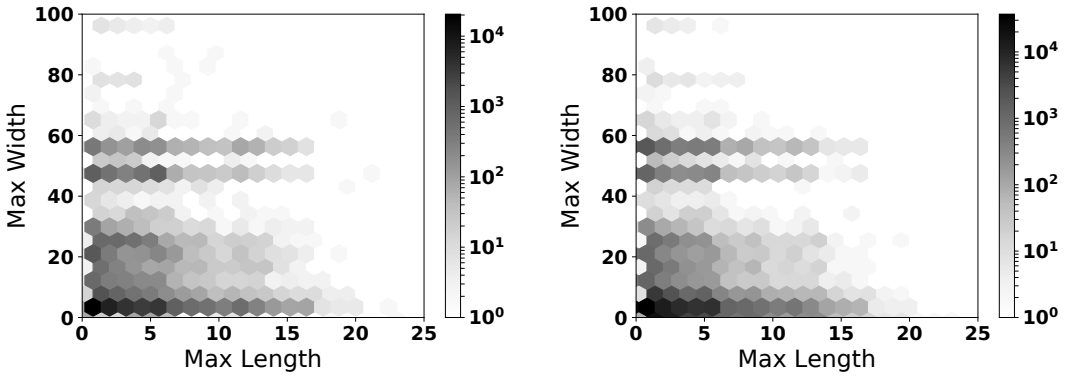
(a) Measured (b) Distinct

Figure 3.10: Maximum length and maximum width

Lite Paris Traceroute during two weeks, starting on 3 April 2018. For each trace, we obtained IP level output and router level output.

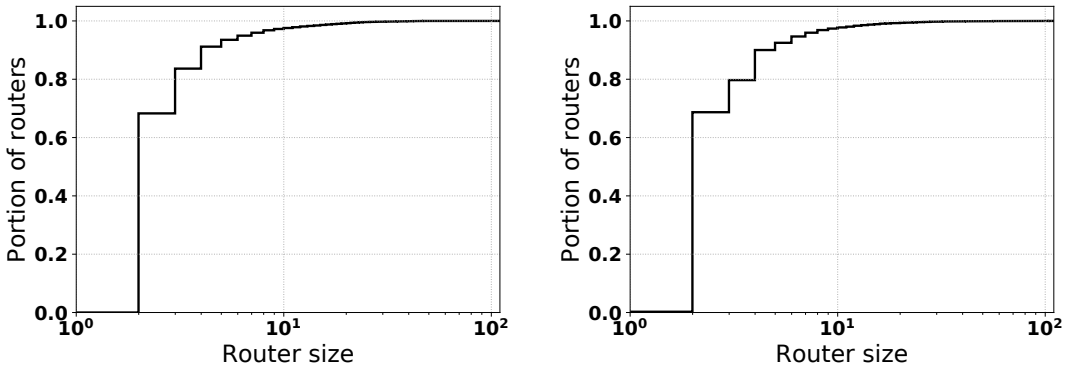
We found 646 cases of distinct address sets (0.98% of the total alias set) that were considered as aliases by one measurement, but discarded or not found by another, although they had both seen the entire address set at the IP level. A deeper analysis showed that 295 of those cases were due to a constant 0 IP ID series collected by one measurement for at least one address in the address set, whereas the other measurement could build a monotonic IP ID time series for each of the addresses in the address set. The remaining 351 cases were false positives, which were then discarded from the router dataset analysed in this section.

We looked at what we term the "size" of the routers that were found, the size being the number of IP interfaces identified as belonging to a router. A route trace from a given vantage point is bound to pick up mostly the ingress interfaces facing that point, which tend to be the ones from which it receives responses, and so this metric will be an underestimate of the true number of interfaces. We also aggregated the IP interface sets from multiple traces through transitive closure based upon two sets having at least one address in common, which may give less of an underestimate, but is still incomplete, as we do not perform full alias resolution on the overall IP addresses set found. CDFs of the sizes are shown in Fig. 3.12. 68% of the routers had a size of 2 and 97% had a size of 10 or less. We found 1 distinct router with more than 50 interfaces, and 5 such routers when



(a) Measured (b) Distinct

Figure 3.11: Maximum length and maximum width joint distributions



(a) Distinct routers (b) Aggregated routers

Figure 3.12: Router size

we aggregated the address sets.

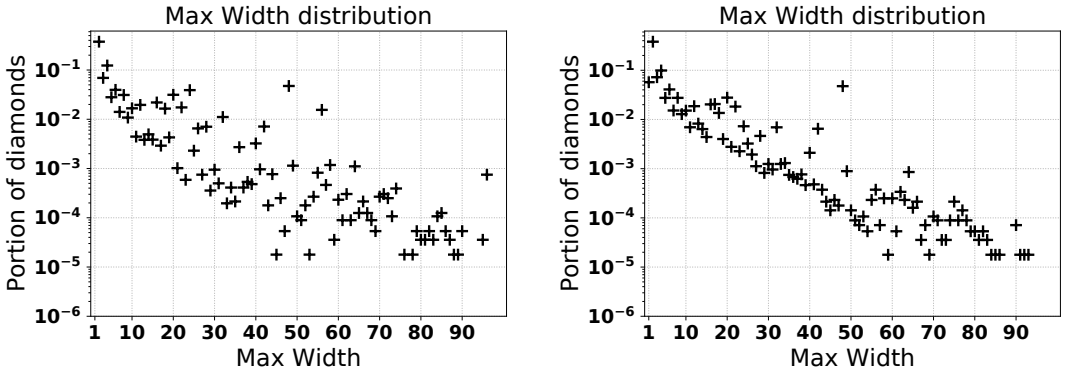
We looked at what happens to each IP level diamond when it is resolved into a router level diamond. There are four possibilities: (1) there is no alias resolution, so the diamond remains the same; (2) the diamond resolves into a single smaller diamond; (3) the diamond resolves into a series of smaller diamonds; (4) the diamond disappears completely, being resolved into a straight path of routers. As Table 3.3 shows, some degree of router resolution takes place on 41.9% of unique diamonds. In comparison, Marchetta et al. [100] saw, in 2016, a 33% reduction in diamond max-width, when applying MIDAR a posteriori to multipath route traces.

We looked at the effect of alias resolution on diamond width. Fig. 3.13 plots the distributions obtained by the MDA-Lite before and after alias resolution. We observe that the peak at maximum width 48 has remained, whereas the one at 56 has disappeared. On closer inspection, we find that the max width 56 diamond at the IP level resolved into several smaller diamonds at the router level. These router-level diamonds were of unaggregated sizes between 2 and 49 IPs.

Finally, we looked at width reduction diamond by diamond. Fig. 3.14 plots the joint

Case	Fraction
No change	0.579
Single smaller diamond	0.355
Multiple smaller diamonds	0.006
One path (no diamond)	0.058

Table 3.3: Effect of alias resolution on unique diamonds



(a) IP level

(b) Router level

Figure 3.13: Maximum width of unique diamonds

distribution of maximum width before and after alias resolution of those diamonds that changed size. Large width reductions are rare, but do take place. The darker grey vertical series of values just to the left of 60 show the maximum width 56 diamonds being broken down into smaller diamonds at the router level.

3.6 Related work

Chapter 2 already described related work on Internet cartography, which MDA-Lite and multilevel route tracing contributions are directly related with.

For the other contributions, Fakeroute is a network simulator purpose-built for one thing: statistical validation of multipath route detection algorithm implementations on a variety of topologies. As such, it does not implement any other features that general network simulators such as ns-3 [77], or emulators, such as GNS3 [9], that can run real router OSes, might offer.

The surveys follow on previous surveys of load balanced paths in the Internet. Our survey provides an update on Augustin et al.'s survey [42] from ten years ago. Like in Marchetta et al.'s survey [100], our survey transforms IP level traces into router-level topologies, but it does so with a single tool, while tracing, rather than with additional measurements by other tools a posteriori. Almeida et al. characterized multipath routes in the IPv6 Internet [35], while we have yet to extend our tool to do the same. Marchetta et al.'s and Almeida et al.'s surveys are quite recent, from 2016 and 2017 respectively, but they report only a maximum of 16 interfaces at a given hop, whereas our survey reveals

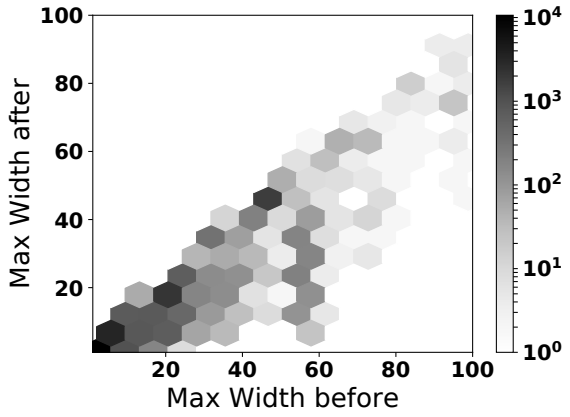


Figure 3.14: Joint distribution of maximum width before and after alias resolution

up to 96.

3.7 Conclusion and future work

Multilevel MDA-Lite Paris Traceroute, makes four contributions related to multipath tracing, each of which can be developed further. (1) For MDA-Lite, the alternative to the Multipath Discovery Algorithm (MDA) that significantly reduces overhead while maintaining a low failure probability, we hope to deepen the mathematical analysis in order to determine significance levels for the results, as had been done for the MDA [126]. Also, the assumption of uniform load balancing, which we believe to hold in almost all cases, could be tested by a rigorous survey, and the algorithm adjusted, which we believe would be straightforward, to take into account uneven load balancing if necessary. (2) For Fakeroute, the simulator that enables validation of a multipath tracing software tool's adherence to its claimed failure probability bounds, we could extend it to simulate exceptions to the assumptions made by the MDA and MDA-Lite. Some assumptions, such as that every probe will receive a reply, often do not hold in practice. Indeed, ICMP rate limiting is one common cause of a lack of replies, and a simulator that takes rate limiting into account could help in designing an algorithm to probe in ways less likely to trigger rate limiting. Another extension might be to allow simulation of multilevel route tracing. (3) For multilevel multipath route tracing, which has provided, a router-level view of multipath routes, we continue to investigate the differences between direct and indirect probing for alias resolution. (4) For the surveys of multipath routing in the Internet, showing, among other things, that load balancing topologies have increased in size well beyond what has been previously reported as recently as 2016, we would like to repeat them, conducting at a larger scale some of the side-by-side comparisons of MDA and MDA-Lite that we have so far conducted only on a smaller scale. Overall, the work is currently entirely focused on IPv4 and can benefit from being applied to IPv6.

This chapter presented Multilevel MDA-Lite Paris Traceroute, a new tool which improves network debugging and troubleshooting, by reducing the number of packets sent, and by embedding in a single command line the router level view of a multipath topology. In addition, MMLPT provides new insights about Internet multipath topology evolution.

For the last decade, load balanced paths have become more complex than ever. Our objective, from this observation, is to build a system that could capture this load balanced paths at Internet scale. The next chapter presents this system.

Chapter 4

Internet Scale multipath tracing

Chapter 3 introduced a new tool that revealed a significant increase of complexity of load balanced topologies in the Internet in the last decade. However, one can argue that it was not representative of the whole Internet, as we randomly selected 350,000 destinations that we divided across 35 vantage points (10,000 destinations per vantage point). The reason for this sampling is that an exhaustive snapshot of the load balanced paths is not achievable in a reasonable time with the existing mapping techniques. A back-of-the-envelope estimate for scanning all the /24 prefixes, given that our survey in chapter 3 took 2 weeks, gives a time of 64.3 years.

This motivates the work presented in this chapter: building a system that can exhaustively map all the load balanced paths of the IPv4 Internet while providing statistical guarantees. This chapter presents D-Miner, a system that reduces the time to complete a snapshot of the entire Internet with multipath tracing to 2.5 days. This work has been published in the proceedings of the *USENIX Symposium on Networked Systems Design and Implementation (NSDI) 2020*.

4.1 Introduction

As we have seen in chapter 3, the state-of-the-art techniques for multipath tracing can not perform exhaustive snapshots of the Internet. As a result, today's IP and router topology snapshots, e.g., [49, 23], incompletely represent the true, multipath, complex forwarding topology. Indeed, our measurements discover >2.7M more edges (64%) in the topology as compared to current state-of-the-art, and significantly more complex topologies than previously reported, including >5k edges in a single provider's topology corresponding to a single /24 destination prefix that they advertise.

The Internet measurement community has long aimed to capture a complete topology of the Internet, especially as it has become clear that partial topologies can lead to faulty conclusions about the network's properties [131, 33, 58]. Obtaining not just accurate, but complete topologies is crucial to understanding the network's resilience to outages and attacks [122, 92], as well as aiding in the conception of applications ranging from content distribution to network security [133]. Further, these topologies play a vital supporting role in other measurement inferences, for instance determining AS relationships, mapping inter-domain congestion [90], and geolocation [84] to name a few.

We developed D-Miner to gather Internet-wide multipath topology maps. D-Miner is

a system that marries two recent advancements in active topology discovery: i) high-speed randomized probing techniques [46] and ii) multipath detection algorithms [41]. At its core, D-Miner rapidly sends probes in a randomly permuted order to avoid overloading any single path, and iteratively completes its view of the network. Whereas prior multipath discovery approaches perform a stateful breadth-first search path exploration to attain confidence in the degree of load balancing along the path, D-Miner decouples probing from statistical inference thereby enabling probe randomization and high-rate probing, while amortizing total probing cost. D-Miner maintains a set of "unresolved" nodes – nodes for which the degree of load balancing is uncertain – and proceeds in rounds until the topology is, statistically, complete (Sec. 5.3). Having implemented D-Miner, we evaluate its performance and overhead (Sec. 5.4). Together, these techniques enable, for the first time, *scalable* multipath topology discovery and Internet-wide mapping.

Using complete Internet-wide snapshots collected using D-Miner, we scrutinize dynamics and topological changes. We find that many "routing changes" are instead due to "load balancer remapping" events – where the load balancer changes its current mapping between flow identifiers and paths. We precisely categorize these events and measure their prevalence, finding that they are a much larger and frequent contributor of path changes than previously recognized (Sec. 4.5).

Finally, we deployed D-Miner and collected multiple topology snapshots over a one-week period in August, 2019. From these snapshots, we characterize the prevalence, size, extent, and location of load-balancing on the modern Internet (Sec. 4.6). This chapter makes four contributions:

1. Development and evaluation of D-Miner, a novel scalable active multipath topology discovery system.
2. Deployment of D-Miner to gather the first Internet-wide IP-level topology snapshots inclusive of load balancing.
3. Detailed characterization of Internet dynamics, including a taxonomy of load balancer remapping events and their extent and prevalence.
4. Public release of D-Miner's code and survey results [7].

4.2 Complementary background and related work

Load balancing, MDA, Yarrp, and Internet mapping systems are concepts that underlie the rest of the technical content of the chapter. These concepts have been explained in details in Chapter 2.

In this section, we provide additional insights about the MDA to explain why it can not be run at Internet scale (Sec. 4.2.1). Then, we further explore the related work on the interesting results found by D-Miner. In particular, we situate them according to previous work on load balancing survey (Sec. 4.2.2, and reappraise previous results on Internet path dynamics based on our findings on load balancer "remappping".

4.2.1 MDA limitations

Sec. 2.1.1 and 3.2.1 detailed the functioning of the MDA. As a remainder, for each node, the MDA computes the minimum number of probes to send to achieve a given failure

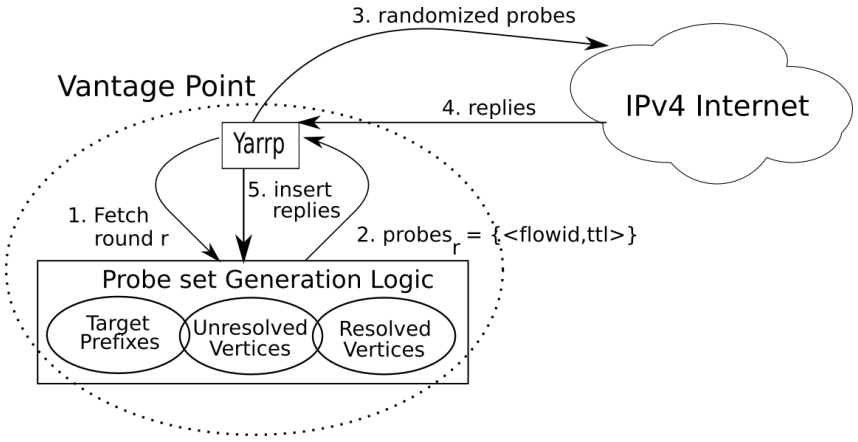


Figure 4.1: D-Miner high-level conceptual overview

probability that it has found all the successors of a node. There are two limitations that prevents the usage of this algorithm at Internet scale. First, the MDA proceeds TTL per TTL, node by node, to send the exact minimum amount of probes to achieve the desired failure probability. This prevents concurrent probing of all the TTLs, therefore linearly increasing the probing time. Second, it maintains, for each node, in addition to the information necessary to match the probes with the replies (see Sec. 2.2.1, Yarrp), its number of successors and the number of replies that it received from this node and its successors. These information can not be maintained for millions of nodes in memory. D-Miner addresses these two limitations. First it modifies the MDA to be able to probe concurrently all the TTLs and all the destinations, for a controlled overhead (Sec. 4.4.2). Then, it uses Yarrp technique to encode all the matching information in the probes. Finally, it stores the necessary information to compute the number of probes needed to achieve statistical guarantees relative to each node in a database, so that the computation can be done *a posteriori* of the probing, providing an adaptive system in terms of memory usage (Sec. 4.3.2).

4.2.2 Load balancing surveys

Our Internet scale survey in Sec. 4.6 confirms two previous results of Augustin et al. [42] and Vermeulen et al. [129]: (1) load balancing is prevalent in the network, as 64.7% of our traces from a source to any /24 prefix contained at least one load balancing router (branching point); and (2) non-deterministic load balancing is rare, with only 1.9% of the branching points identified as implementing this behavior.

4.2.3 Internet path dynamics

D-Miner enables researchers, for the first time, to obtain snapshots of the entire Internet inclusive of all load-balanced paths. This new and more complete view of the topology allows D-Miner to expand the results on load balancing characterization at Internet scale. Augustin et al. made the first study of load-balanced paths a decade ago, finding topologies having up to 16 such paths [42]. More recently, Vermeulen et al. have shown that per-flow load-balanced paths have become more complex, by finding topologies with up

to 92 interfaces at the same TTL [129]. In both of those studies, the set of destination prefixes was a subset of the entire Internet, with, respectively, ∼120k and ∼350k targets. In contrast our survey contains traces to ∼14.4M /24 destination prefixes.

D-Miner allows us to provide new insights into Internet dynamics induced by load balancing. In Sec. 4.5.2, we show that a "routing change" is more nuanced that previously understood, and that such changes can be due in part to the remapping behavior of production load balancers. Paxson's canonical work on Internet dynamics [108] studied *persistence* and *prevalence* of routes. Given a source/destination pair, persistence characterizes the number of different routes observed across time between this pair. The prevalence of a route defines if, within the set of routes that have been observed, one is more dominant than another. The main result was that Internet routes were globally stable across time. Note that this work was performed two decades ago and did not take load balancing into account. Almost a decade ago, Cunha et al. reappraised Paxon's work in light of load balancing [61], and found that Paxson's results still held. They also stated that load balancing remapping is infrequent. We show that it is a widespread phenomenon in today's Internet.

More recently Cunha et al. developed DTrack [62], a system that maintains an inferred topology and attempts to detect and predict path changes, although such prediction is difficult. Our work helps provide insight into potential root causes of this prediction difficulty.

4.3 Algorithm

D-Miner is designed to capture Internet topology snapshots inclusive of all load-balanced paths. At its heart, D-Miner uses Yarrp's randomized and stateless probing to achieve high probing rates. To this, it adds probe set generation logic that keeps track, on a per-node basis, of whether all outbound load-balanced edges have been discovered with high probability. The logic guides Yarrp through multiple rounds until the full discovery criterion has been satisfied for almost all nodes.

See Figure 4.1 for a high-level schematic of D-Miner. Yarrp and the probe set generation logic are deployed at a single vantage point from which Yarrp probes a set of target prefixes in the IPv4 Internet. D-Miner proceeds in rounds, maintaining sets of "resolved" and "unresolved" vertices. Resolved vertices correspond to nodes where all outgoing load balanced links have been discovered with high probability toward the set of target prefixes. Conversely, unresolved vertices require further probing to ascertain whether any load balanced edges emergent from a node have not yet been discovered.

D-Miner's main steps are: (1) Yarrp requests the set of probes for round r; (2) the probe set generation logic returns the `<flow ID,TTL>` pairs that correspond to the current set of unresolved vertices; (3) these pairs are randomized and probes are sent at high speed using the Yarrp technique; (4) as replies return, they are processed by Yarrp; and (5) they are used to update the set of known nodes and each node's state as either an unresolved or a resolved vertex. Rounds continue until 99% of the target prefixes have been resolved.

Whereas Yarrp is totally stateless, D-Miner requires state to be retained from round to round. A key challenge is to manage that state in a manner that does not diminish Yarrp's performance. Our solution is discussed in Sec. 4.3.2.

4.3.1 Bootstrapping D-Miner

Since D-Miner is guided by the set of unresolved vertices, it requires a boostrapping round to seed the set. This round is a slightly modified version of a classic Yarrp snapshot of the IPv4 Internet. Whereas Yarrp sends one probe packet per TTL to each /24 prefix, with the exception of the private and reserved IPv4 prefixes defined by RFC 6890 [60], D-Miner sends $n_1 = 6$ probes per /24, each with a different flow identifier. This number comes from the MDA stopping condition for 0.05 failure probability, described in Sec. 4.2.1, that there is just a single node at a given TTL when probing towards a given destination. The six flow identifiers correspond to the six first destinations in the /24. The /24 granularity corresponds to the commonly accepted longest BGP prefix [121].

In the classic MDA, the n_1 packets all have a common destination, however D-Miner varies the flow identifier by varying the destination within the target prefix. This allows it to find per-destination-prefix load balanced paths in addition to the per-flow load balanced paths that classic MDA finds.

As stated in Sec. 4.2, Yarrp uses a pseudo random permutation of 32 bits to determine the parameters for each successive probe: the first 24 bits determine the /24 destination prefix and the first 5 bits of the remaining byte determine the TTL. This leaves 3 bits, which is sufficient for D-Miner to select $n_1 = 6$ different addresses within the destination prefix.

4.3.2 Maintaining State

Yarrp encodes the originating TTL, timestamp, and checksum in the probe header fields. These values are visible in the quotations that arrive in the probe replies, allowing Yarrp to reconstruct the probe that generated a particular ICMP without maintaining any internal state. While each individual Yarrp probing round is stateless, D-Miner must maintain state from round to round in order to keep track of which vertices have been resolved and which ones remain unresolved. To this end, D-Miner extracts the following data from each reply: the original probe's source and destination IP addresses, port numbers, and original TTL; and the reply's source IP address and ICMP type and code.

So that D-Miner could obtain rapid results for complex queries on tables of billions of rows, we sought a database system that is optimized for online analytical processing, settling on ClickHouse [5]. The data from each reply is inserted and ordered by: source IP address, /24 destination prefix, destination IP address, TTL, source port number, and destination port number. ClickHouse is highly parallelized and its `groupArray` features make the algorithm calculations described in Sec. 4.3.3 and the analyses of Sec. 5.4, Sec. 4.5, and Sec. 4.6 tractable.

4.3.3 Subsequent probing round computation

Once the replies have been inserted into the database, we query it to generate the next round of probes. Our goal, conceptually, is to calculate the set of additional probes with new flow identifiers required to meet the remaining statistical guarantees for each /24 prefix, given the current knowledge of the topology. Mathematically, the next round probes is the minimal expected set of probes needed to reach the statistical guarantees for each branching point, grouped by /24 prefix.

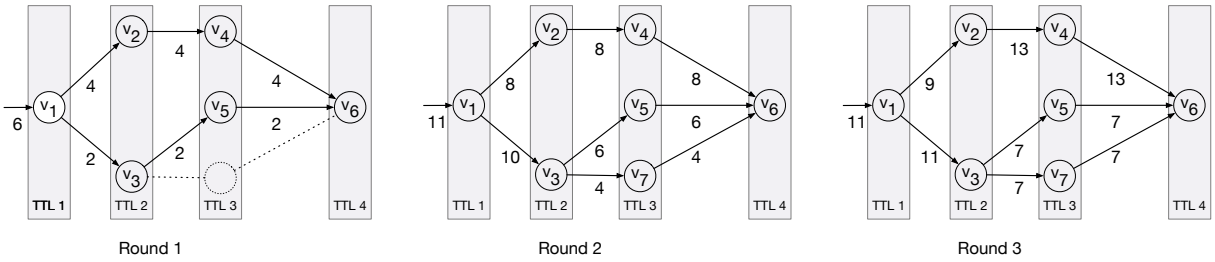


Figure 4.2: Example topology resolved by three rounds of D-Miner probing. Link annotations represent number of probes expiring at ingress interface of subsequent node.

Reducing MDA statefulness

This section describes our algorithm to generate a new batch of probes given a topology and the set of probes already sent.

Let us fix a source and a /24 destination prefix. We present an example in Figure 4.2 to help provide intuition. This topology illustrates a possible result after each of three hypothetical rounds of probing. Each link is annotated with the number of probes that expired at the ingress interface of the subsequent node. At round 1, D-Miner sent $n_1 = 6$ probe packets per TTL, each with varying destinations in the destination prefix to vary the flow identifier. The value of n_1 is determined by the desired failure probability to find all the successors of a branching point. In this work we set $n_1 = 6$, corresponding to a failure probability of 0.05, which is the default value used by the MDA implementation in previous work [42, 62].

Recall, after sending 6 probes and only discovering a single successor, we have a probability of 0.95 that there is indeed only a single successor ($n_1 = 6$), while we must send 11 probes to achieve the same probability that there are only two successors ($n_2 = 11$). To understand how we compute the next batch of probes, we introduce the following notation:

Let us fix the TTL to h.

Let R_h be the set of nodes discovered at TTL h that have not been resolved yet.

Let D_h be the probability distribution of nodes responding for TTL h after the current probing round. For example, in Figure 4.2 after the first round of six probes per TTL, $D_2 = \left\{\mathrm{v}_2 = \frac{4}{6}, \mathrm{v}_3 = \frac{2}{6}\right\}$.

Let k_v be the number of successors for node v.

Let t_h be the number of probes already sent at TTL h.

Let n_k be the stopping point described in Sec. 4.2.1.

Proposition 1. *Given h, R_h and D_h, the minimal expected number of probes needed to reach MDA statistical guarantees for all the elements of R_h is:*

$$\max_{v \in R_h} \left(\frac{n_{k_v}}{D_h(v)} - t_h \right) \qquad \text{At TTL } h$$

$$\max_{v \in R_h} \left(\frac{n_{k_v}}{D_h(v)} - t_{h+1} \right) \qquad \text{At TTL } h+1$$

Proof. Let $v \in R_h$. The MDA hypothesis, which is that v might have a $(k_v+1)^{\text{th}}$ successor tells us that we need to send n_{k_v} probe packets with TTL h that first reach v, and then send n_{k_v} with the same flow identifiers to TTL $h+1$. Let X_v be the random variable representing the number of probes that reach node v given D_h. Our objective is to find the minimum number of probes N such that:

$$\forall v, E[X_v] = D_h(v)N >= n_{k_v} \tag{4.1}$$

For this condition to hold, we must set N at minimum to:

$$N = \max_{v \in R_h} \frac{n_{k_v}}{D_h(v)}$$

We have:

$$\forall v, E[X_v] = D_h(v) \max_{v' \in R_h} \frac{n_{k_{v'}}}{D_h(v')} >= D_h(v)\frac{n_{k_v}}{D_h(v)} = n_{k_v} \tag{4.2}$$

We just subtract the number t of probes that we have already sent to TTL h, and this concludes the proof. □

Every TTL h generates a number of additional probes for TTL h and TTL $h+1$. For each TTL h, we therefore have two possible values: the one generated by additional probes for TTL $h-1$ and the one generated by additional probes for TTL h. So that the condition given in Eq. 4.1 holds for every node of the topology, we choose the maximum between these two values (rounding fractional values to integers).

Let us perform the numerical application on the topology of Figure 4.2. After round 1, we have discovered the topology on the left side of the figure. At TTL1, Node v_1 has two successors, $D_1(v_1) = 1$. 6 probes have been sent to TTL 1 and 6 probes have been sent to TTL 2. Proposition 1 brings $n_2 - 6 = 5$ additional probes for TTL 1 and 2. At TTL 2, Node v_2 has one successor, and $D_2(v_2) = \frac{2}{3}$. Node v_3 has one successor, and $D_2(v_3) = \frac{1}{3}$. 6 probes have been sent to TTL 2 and 6 probes have been sent to TTL 3. Proposition 1 brings $3n_1 - 6 = 12$ additional probes for TTL 2 and 3. At TTL 3, notice that v_4 and v_5 are similar to v_2 and v_3, so that Proposition 1 brings $3n_1 - 6 = 12$ additional probes for TTL 3 and TTL 4. For each TTL, we take the maximum number of probes between TTL and TTL-1. At the end, we have to send for the second round: 5 probes at TTL 1, 12 at TTL 2, 12 at TTL 3 and 12 at TTL 4.

The figure in the center shows the state of the topology after round 2. At TTL 1, we see that v_1 has been resolved because it has two successors and $11 = n_2$ flows pass through it. At TTL 2, v_2 has also been resolved because it has one successor and $8 \geq n_1$ flows have been sent through it. v_3 is not solved, because now it has two successors so it needs $n_2 = 11$ flows that pass through it. We have $D_2(v_3) = \frac{10}{18}$ and 18 probes have been sent to TTL 2 and 3. Proposition 1 brings $\frac{18}{10}n_2 - 18 = \lceil 1.8 \rceil = 2$ additional probes for TTL 2 and 3. At TTL 3, we see that v_4 and v_5 have been resolved, but v_7 has not. We have $D_3(v_7) = \frac{4}{18}$ and 18 probes have been sent to TTL 3 and 4. Proposition 1 brings $\frac{4}{18}n_1 - 18 = 9$ additional probes for TTL 3 and 4. For each TTL, we take take the maximum number of probes between TTL and TTL-1. At the end, we have to send for the third round: 0 probe at TTL 1, 2 at TTL 2, 9 at TTL 3 and 9 at TTL 4.

The figure on the right shows the state of the topology after round 3. We see that now v_3 and v_7 have been resolved, so that we consider the entire topology as resolved.

Varying the flow identifier

For each destination prefix and TTL where additional probes are needed, we select new flow identifiers by incrementing the last destination in the prefix used in the previous round. In the example of Figure 4.2, suppose that our prefix is X.Y.Z.0/24. After round 1, the first 6 IPs of the prefix have already been used as probe destinations, so we would send 5 more probes at TTL 1 from X.Y.Z.7 to X.Y.Z.11, 12 more probes at TTL 2 from X.Y.Z.7 to X.Y.Z.18, etc. If there are no more destinations available in the prefix, we change the destination port to vary the flow identifier.

4.3.4 Randomizing the probes

Randomizing the probes is done per-flow. For each prefix which needs additional probes sent, we group the additional probes by flow. In the example of Figure 4.2, after round 1, we send X.Y.Z.7 at TTL 1, TTL 2, and TTL 3, and TTL 4. All of these are packed together so that we ensure that probes with the same flow identifier are sent in a very short time window. This avoids the inference of false links due to potential routing changes during the probing time. Nevertheless, it is possible that X.Y.Z.8 probes are sent at a separate time from X.Y.Z.7. In this way we retain one of the benefits of Yarrp's randomization, to minimize potential ICMP rate limiting.

4.3.5 Per-Packet load balancing

As described in Sec. 4.3.1, the first round of probes allows us to discover if there is one or more load-balancers on the path from the vantage point to the destination prefix. However, sending only one packet per flow identifier does not reveal the nature of the load balancing, i.e., deterministic (per-destination or per-flow) or non deterministic (per packet). For per-packet load balancers, the links between interfaces of a traceroute can not be reliably inferred. Because we want to be able to flag per packet load balancers, D-Miner sends two back-to-back probes per flow instead of one until it reaches a defined threshold probability that the branching point is not a per packet load balancer. If we get two different responses for flows with the same flow identifier, the branching point is flagged as a per packet load balancer and the edges discovered after it are ignored in the results. Mathematically, an upper bound of the probability to miss that a load balancer is actually a per packet load balancer is the probability that all the pairs of probes with the same flow identifiers passing through it get the same reply IP. Suppose there is a branching point with k branches. Then the probability that all these pairs of probes get the same response is then $\frac{1}{k^p}$ where p is the number of pairs of probes sent going through this branching point. We set the threshold probability in D-Miner to 0.95, as previously done in [42].

4.4 Evaluation

Over one third of the edges in the Internet's topology are not being revealed by current state-of-the-art methods, as Sec. 4.4.4 describes. D-Miner, run from from six PlanetLab Europe vantage points, discovered more than 7.1 million edges during a week of August 2019, a time during which Ark, probing from its 112 VPs, found ∼4.4 million edges and

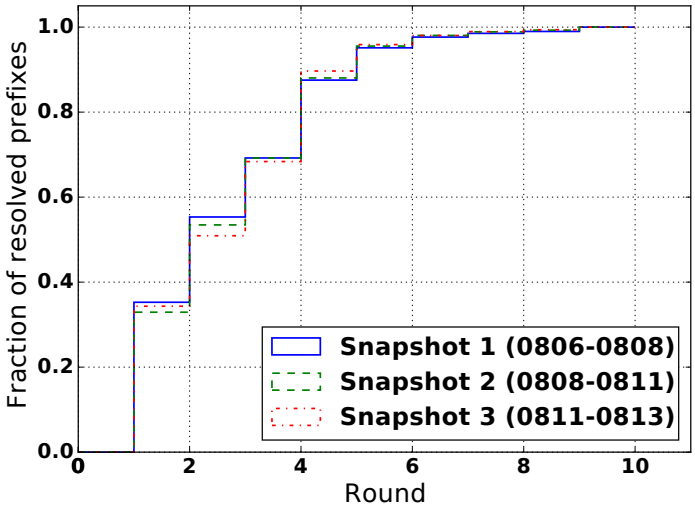


Figure 4.3: CDF of /24 IPv4 prefixes meeting their statistical guarantees, over three 10-round snapshots (dates in parens)

Yarrp, probing from the same PLE VPs as D-Miner, found ∼2.5 million. Although this additional coverage comes at the cost of 2-4 times as many probes compared to these existing approaches (Sec. 4.4.4), we show that this volume is required due to forwarding path dynamics.

In addition to evaluating node and edge discovery, we evaluate D-Miner's algorithmic and system properties. D-Miner's round-iterative design engenders 14% higher overhead than would be incurred sending traditional source-to-destination style MDA Paris Traceroutes from the same vantage point, as Sec. 4.4.2 shows. This is the cost associated with D-Miner adopting a stateless and randomized (Yarrp-based) design, in exchange for the advantages conferred by this design in terms of probing speed. Sec. 4.4.3 evaluates D-Miner's running time. First, however, Sec. 4.4.1 describes the datasets we collected, and shows that 10 rounds suffice to reach statistical guarantees for 99% of the /24 destination prefixes.

4.4.1 Dataset

Our evaluation dataset includes three D-Miner snapshots, each consisting of 10 rounds. These snapshots were collected over a one-week period, from 6-13 August 2019, using a single vantage point at our university, which enjoys a 1 GB link to the Internet. Each snapshot was collected using a probing rate of 100,000 pps; this rate was capped out of respect for network and service provider policy concerns; D-Miner is capable of probing much faster ($>$ 800,000 pps observed).

In addition, for the topology discovery section, we have deployed D-Miner on six PlanetLab Europe (PLE) [19] nodes located in Europe and run it at a lower rate, 10,000 pps, during the same week. We used UDP probes as these have been shown to discover more links than other protocols [98]. A web page hosted at the IP address of our vantage point described the experiment and provided instructions for opting out; we did not receive any such requests during our measurements.

Figure 4.3 illustrates how 10 rounds of probing suffices to achieve statistical guarantees

Overhead Factor	**Snapshot 1** (Aug 6-8)	**Snapshot 2** (Aug 8-11)	**Snapshot 3** (Aug 11-13)
Loss (I)	481,359,024	448,428,279	569,173,354
Reached destination (II)	166,941,456	165,227,106	163,821,372
Stateless probing (III)	54,023,123	46,324,355	50,012,378
Total Overhead [pkts]	**702,323,603**	**659,979,740**	**783,007,104**
D-Miner Sent [pkts]	6,976,307,081	6,569,819,008	6,598,837,985
MDA Sent [pkts]	6,273,983,478	5,909,839,268	5,815,830,881
D-Miner Overhead [%]	**12**	**12**	**14**

Table 4.1: Probing Overhead of D-Miner versus MDA.

toward more than 99% of the IPv4 /24 destination prefixes. The mean portion of resolved prefixes over the three 10-round snapshots is 99.6%.

4.4.2 Probing overhead

Intuitively, we might expect D-Miner to have lower overhead in order to achieve scale. However, the stateless high-rate probing required comes at the cost of additional total probing. The probing overhead generated by D-Miner compared to sequentially running the traditional MDA to all of the /24 prefixes depends on three factors: (I) potential loss induced by a high probing rate; (II) the TTL at which the MDA would have stopped because of reaching the destination; and (III) sending more probes than required to reach the statistical guarantees due to stateless probing. Table 4.1 quantifies each of these overhead factors across our three snapshots, where we find a maximum overhead of 14%.

Note that we conservatively compute the loss due to high probing rates. If for a given (destination prefix, TTL) pair, if at least one of the probes receives one response, we count all the probes sent with the same (destination prefix, TTL) pair as losses if they do not produce a response. Conversely, if no responses at all are received for this (destination prefix, TTL) pair, i.e., this hop is "anonymous"[1], we do not count these probes as lost.

To compute the TTL at which the MDA would have stopped, we find the minimum TTL for which all probes' reply IPs equal their destination IP. If we never receive a reply from the destination, we assume that MDA would act like a default linux traceroute, i.e sending probes until it reaches the maximum TTL (30).

And finally, to compute the stateless overhead, we simulate for each of the destination prefixes a run of the MDA with the same flow identifiers and compute the actual number of probes at which it stops due to having reached the statistical guarantees.

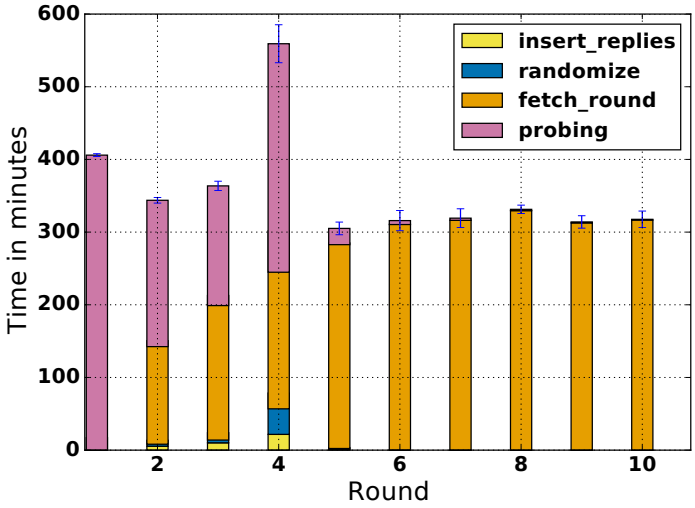


Figure 4.4: Time spent in each step of D-Miner for each round, averaged across three snapshots.

4.4.3 Run time

The server used for the computation was the same as the one we used for probing, provisioned with 16 cores and 187 GB of RAM. Figure 4.4 shows the stacked error bars with standard deviation of the time spent across each round in each routine. The sum over the rounds indicates that a snapshot of 10 rounds takes an average of 3,713 minutes (about 2 days and 14 hours) to complete.

We distinguish two phases in our system: Until round 4, the `probing` routine consumes a significant portion of the total round time, however, in rounds 5–10, almost all the time is spent in the `fetch_round` routine. Note the relationship between time (Figure 4.4) and the fraction of resolved prefixes (Figure 4.3). While the amount of probing time required in rounds 6-10 is relatively inexpensive as compared to the earlier rounds, we see that the algorithm is primarily optimizing at this point, with >90% of the prefixes resolved by the fifth round. Conversely, only ∼50% of the prefixes are resolved after the third round, demonstrating that the runtime cost of probing in the early rounds is required.

Notice the evolution of the time spent in the `fetch_round` routine: The time needed to compute the next round probes increases with the size of the table of our database in rounds 1–4. Then, from round 5 on, the reduction in time spent `probing` indicates that far fewer probes are sent, and consequently the table does not grow as much as during the earlier rounds. Still, the time of `fetch_round` remains constant; one must recompute the state of each branching point at each round to compute the next one.

4.4.4 Topology discovery

Finally, we consider the topology discovery results themselves. Note we ignore responsive target addresses (since we are only interested in the routed topology) as well as any edges connecting the destination nodes.

Figure 4.5 shows the cumulative discovery, with error bars with standard deviation,

[1]Represented as a * in traditional traceroute output[25]

	Nodes				Edges			
	Snapshot 1 (Aug 6-8)	**Snapshot 2** (Aug 8-11)	**Snapshot 3** (Aug 11-13)	**Agg.**	**Snapshot 1** (Aug 6-8)	**Snapshot 2** (Aug 8-11)	**Snapshot 3** (Aug 11-13)	**Agg.**
D-Miner	1,057,832	1,017,389	1,024,178	1,373,149	2,906,204	2,997,581	3,059,770	4,600,689
Yarrp	545,536	492,694	516,363	632,405	977,201	898,607	916,196	1,279,171
Multi-VP Ark	1,432,604	1,635,779	1,343,009	1,923,038	3,044,747	3,472,120	2,994,190	4,365,515
Multi-VP Yarrp	802,891				2,483,816			
Multi-VP D-Miner	1,613,301				7,143,490			

Table 4.2: Topology discovery

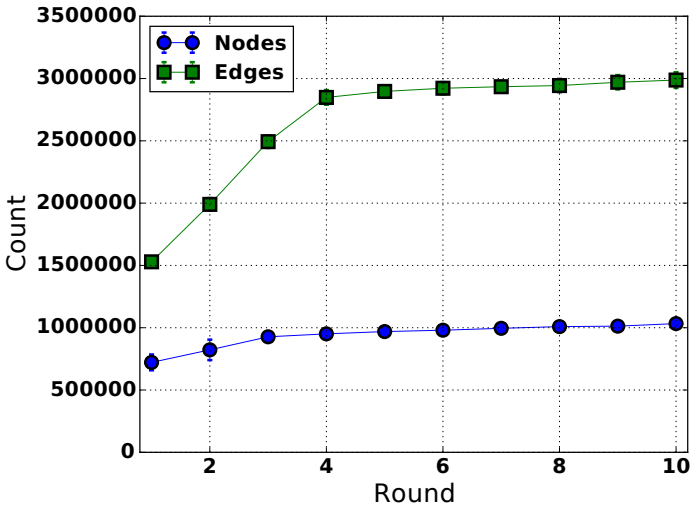


Figure 4.5: Per-round node / edge discovery for three snapshots.

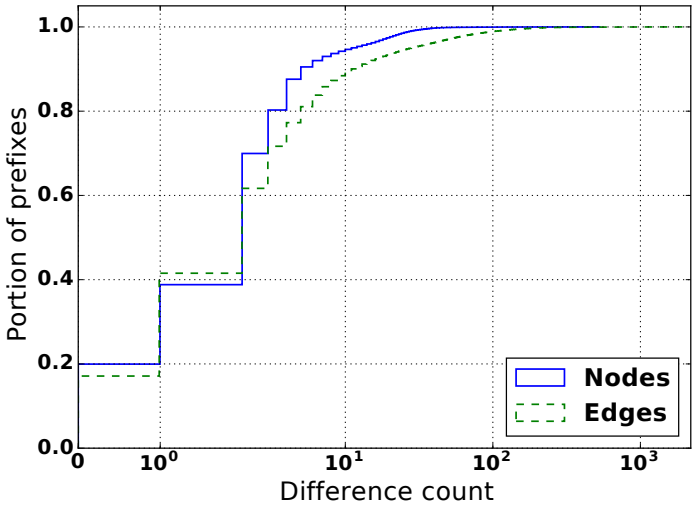


Figure 4.6: CDF of the minimum difference of number of nodes and edges per prefix.

of each round across the three D-Miner snapshots. We find that both nodes and edges have similar behavior with two discovery phases, with an initial high discovery-per-round phase until round 3 (nodes) and 4 (edges) followed by a refinement phase from round 4 (nodes) and 5 (edges) to round 10. Note also that the number of nodes and edges varies little across the three snapshots, however, this does not mean that they discover the same set of nodes and edges as we will discuss.

It is challenging to make direct comparisons previous topology data sets – not only is the network dynamic, but the results are also significantly influenced by differing vantage point(s). Instead, we seek to make a reasonable comparison of D-Miner with existing Yarrp and Ark systems.

To compare with Yarrp, we extract from our D-Miner snapshots results given by selecting only one destination per /24 prefix in the first round. To compare against Ark, we obtain all topology traces from all 112 vantage points corresponding to the date range of our snapshots (this includes 19 "cycles" that were performed during the week). We aggregate the topology found from all cycles during the D-Miner snapshot collection to compare findings from the same time period.

Table 4.2 gives the comparative topology results. The "Agg." column shows the aggregated results over the three snapshots. The two last lines of Table 4.2 show the aggregated topology discovery results across the six PLE nodes for Yarrp and D-Miner over the same week covered by the three snapshots.

The multiple vantage point results in this table are significant. With 6 vantage points, D-Miner discovers >7M edges and approximately 1.6M nodes. To verify that this difference is primarily due to load balancing, we look at how standard Yarrp performs when used from these 6 vantage points. We see that D-Miner discovers two times more nodes and almost three times more edges than Yarrp.

Interestingly, we see that D-Miner from a single vantage point still benefits from snapshot aggregation. This means that there are non-negligible variations in the discovery of the three snapshots. Figure 4.6 shows the minimum per-prefix difference of number of nodes and edges discovered between the three snapshots over the 14,461,947 prefixes that

we probed. As an example, for a prefix, if snapshot 1 discovers 20 nodes and 40 edges, snapshot 2 discovers 20 nodes and 34 edges, and snapshot 3 discovers 20 nodes and 36 edges, the minimum difference for nodes is 0 and 4 for edges. We observe two results: Less than 20% of the prefixes discover the same number of nodes and edges for the three snapshots, and 88.3% of the prefixes have a variation of less than 10 edges.

We believe that these variations are related to load balancing remapping (see Sec. 4.5.2). Indeed, the difference is not attributable to high probing rate induced loss, as Table 4.1 has shown than fewer than 10% of the probes were lost due to rate across the three snapshots.

Please note that we do not claim a comprehensive comparison with Ark discovery. The two systems have intrinsic differences that would not allow us to state conclusively why D-Miner or Ark would discover more than the other system. For example: (1) Ark uses ICMP probes, which has been demonstrated by Luckie et al. in [98] to discover fewer links than UDP. (2) The two systems' vantage points are not located at the same points in the network. Therefore, Ark could miss load balanced paths that are in a region of the Internet that is not accessible from its vantage points.

Probes sent

In total, we compute that Ark sent 5,935,460,660 probes. From Table 4.1 we compute that D-Miner from one vantage point sent 20,144,964,074 probes. And finally, we compute that the multiple vantage points version of D-Miner has sent 13,192,962,692 probes in total across the 6 PLE nodes.

These numbers give an idea of the overhead necessary to discover the load balanced paths. We show in the next section that reducing this number is hard because of the dynamics.

4.4.5 Validation on ground truth

To complement the formal and experimental analysis of our system, we solicited ground truth from operators. Of 380 interfaces with multipath edges discovered within Internet Initiative Japan (IIJ), they validated that 51 interfaces belonging to NTT were on PPPoE routers performing ECMP to IIJ. We further learned that the remaining 329 interfaces are performing ECMP inside IIJ's network.

In addition to direct correspondence with IIJ, we developed a website [31] where operators can validate links discovered by D-Miner. We received responses from three operators, for a total of 20 links. 18 validated correctly while two were declared as false. We re-conducted paris traceroute measurements to the destinations corresponding to the two false positive links. These links were found between the penultimate and the last IP seen in the traces. The last IP, which was not the destination IP, was repeating on all the subsequent TTLs until 30. Our interpretation is that this error was due to a routing loop or misconfiguration.

4.5 Forwarding path dynamics

With D-Miner, we reveal aspects of Internet dynamics that were under estimated [61] by the research community. 28.6% of D-Miner's probes saw their reply's IP address change,

despite the flow identifier remaining constant over our three August 2019 snapshots. This would seem at first glance to be a startlingly high figure, implying more extensive routing changes than one might expect throughout the Internet [108, 61]. But we provide evidence that, rather than routing changes, another phenomenon that we term *load balancer remapping* was responsible for at least 52% of these changes. These observations imply that future work should be cautious in attributing an observed `traceroute` change to a routing change.

4.5.1 Taxonomy of probe changes

A probe is uniquely identified by its (flow ID,TTL) pair. We say that there is a *probe change* if a probe elicits a reply from a different IP address in snapshot $i+1$ than in snapshot i. There may be as many probe changes as there are probes. We distinguish between meaningful and trivial changes.

To begin with, we do not count an *absence of response* as a meaningful change. That is, if a probe elicits a response in snapshot i and there is no reply to the probe in snapshot $i+1$, this difference may be attributable to loss or rate-limiting and is not indicative of a change. Similarly, we ignore absence in one round followed by a response in the next.

We are particularly interested in examining each probe change from the perspective of its predecessor node. Consider a probe (X, h) with flow ID X, sent with TTL h, revealing a vertex v_1; and a probe $(X, h+1)$ revealing a vertex v_2. If, in the next snapshot, (X, h) reveals v_1 again but $(X, h+1)$ reveals v_3, it is reasonable to infer that a mechanism at the router with interface v_1 is responsible for this probe change. Perhaps there has been a routing change, and the routing table at that router has been updated. Or, and this is the possibility that we focus on here: perhaps there has been no routing change, but instead that router is a load balancing router and the probe change results from a new load balancing decision for packets with flow ID X.

Previous research has identified *per-packet load balancing* [42] where a router simply disregards the flow ID X in its load balancing decision, for instance directing some packets to v_2, some to v_3, and some, perhaps, to other neighboring vertices, in a round-robin or (pseudo-)random fashion. As Sec. 4.3.5 describes, D-Miner tests for per-packet load balancing, and we exclude any reply variation due to that mechanism from our accounting of meaningful probe changes.

A possibility that previous literature has not explored is that a probe change could result from a per-flow or a per-destination load balancer making a load balancing change rather than a routing change. As an example, consider probe packets with flow IDs X and Y that both have the same destination IP address d. In snapshots i and $i+1$, probes (X, h) and (Y, h) both elicit replies from v_1, whereas in snapshot i, probe $(X, h+1)$ elicits a reply from v_2 and probe $(Y, h+1)$ from v_3, and in snapshot $i+1$, the replies are reversed. From one snapshot to the next, there has been no routing change for d at v_1: packets with that destination address continue to be load balanced across v_2 and v_3. But suppose the test for per-packet load balancing at v_1 has failed. We are left to consider that the probe change results from an update to the hashing decision that assigns flow IDs to next hop IP addresses. This is what we term *load balancer remapping*.

4.5.2 Load balancer remapping

Production systems are more nuanced in their behavior relative to the overview of load balancing in Sec. 4.2. For deterministic load balancing, in addition to header fields used to compute the hash function, a seed is generally added to avoid a phenomenon called *load balancing polarization* [4, 27, 29], which occurs when load balancers are chained and apply the same hashing algorithm, potentially causing load imbalance.

We experimented with Cisco routers running IOS 12 in a lab environment and confirmed that this seed is configurable. If the Cisco router reboots and has no saved seed, it generates a new random seed. For Juniper and Huawei, documentation tells us that they also use a seed [27, 29].

Thus, at least for Cisco, Juniper, and Huawei routers, a flow that took a certain load balanced path before a reboot can take a very different path after the reboot. Moreover, removing or adding an interface on a load-balancing router(s) toward a destination can cause the path assignment to be recalculated [26, 21]. The particulars of the load balancing implementations in production can cause the reply IP address of a probe to change, *even when no routing change has occurred, and the flow identifier is constant.*

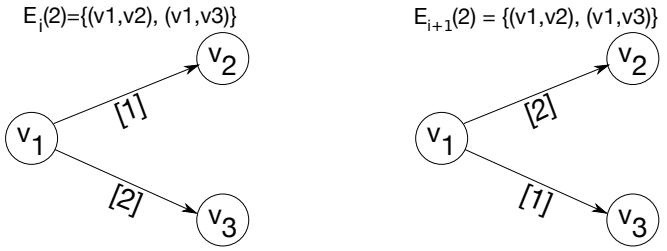
We attempt to identify in our dataset probe changes that correspond specifically to load balancer remapping. Our conservative rule of thumb is to say that if we observe a meaningful change for probe $(X, h+1)$, but at least one of the edges of the predecessor vertex is the same between snapshots, then remapping is taking place. If no edges are the same, it could still be the result of load balancer remapping instead of a routing change, but without evidence to allow us to infer this. Thus, our observations will underestimate the ubiquity of remapping.

We define $E_i(p)$ to be the set of out edges in snapshot i from the predecessor vertex of a meaningful change for probe $p = (X, h)$ (again, where a probe is a flow ID X and TTL h tuple). For each candidate meaningful-probe-change, we infer that the change is due to remapping if $E_i(p) \cap E_{i+1}(p) \neq \emptyset$.

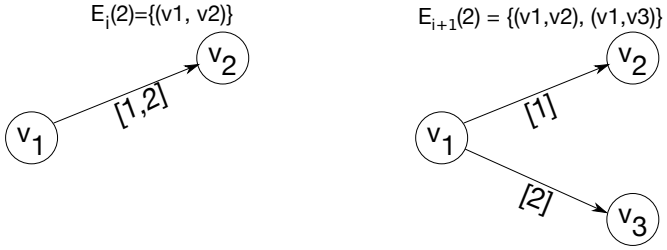
Several classes of meaninful changes can be more more precisely characterized into subcategories. Either $E_i(p)$ and $E_{i+1}(p)$ can be: (1) equals, meaning that the changes might be due to a router reboot ($E_i = E_{i+1}$); (2) one set is included in the other, corresponding to potential addition/removal of some load balanced paths ($E_i \subset E_{i+1}$ or $E_{i+1} \subset E_i$); or (3) the sets have elements in common, but no inclusion relation can be established ($(E_i \subsetneq E_{i+1}) \wedge (E_{i+1} \subsetneq E_i)$), e.g., when load balanced paths are both added and removed.

Figure 4.7 illustrates these three cases by depicting of remapping from one snapshot, i (left), to the next, $i+1$ (right). Consider probe 2 in Fig 4.7a which is a meaningful change since it elicits v_3 in the first snapshot and v_2 in the second. The edges from the predecessor of these changes are the same between snapshots, i.e., $E_i(2) = E_{i+1}(2) = (v_1, v_2), (v_1, v_3)$. Note that the reciprocal change is observed by the flow 1 probe in $i+1$, for a total of two inferred remapping events.

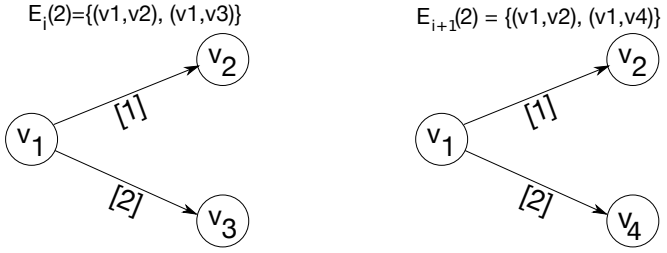
In the example of Figure 4.7b, probe 2 is again a meaningful change, but in this instance elicits a new vertex v_3 in the second snapshot. In this case, the predecessor edges in the first snapshot are a subset of the second, i.e., $E_i(2) = (v_1, v_2)$ and $E_{i+1}(2) = (v_1, v_2), (v_1, v_3)$. Finally, Figure 4.7c shows an example of no inclusion relation where there is both a new and deleted edge due to remapping.



(a) Remapping, no new edges ($E_i = E_{i+1}$)



(b) Remapping, new edge ($E_i \subset E_{i+1}$)



(c) Remapping, new and deleted edges ($E_i \subsetneq E_{i+1}$) $\wedge$ ($E_{i+1} \subsetneq E_i$)

Figure 4.7: Examples of three different types of changes for probe 2 across two snapshots i (left) and $i+1$ (right). In all cases, $E_i \cap E_{i+1} \neq \emptyset \implies$ we infer these as remapping events.

Changes	0	1	2
Count	2,184,277,681	652,241,148	223,614,385
Total	3,060,133,214		

Table 4.3: Changes per (flow ID, TTL) probe on three snapshots.

$E_i \cap E_{i+1} = \emptyset$	$E_i \cap E_{i+1} \neq \emptyset$			
	$E_i = E_{i+1}$	$E_i \subsetneq E_{i+1}$	$E_i \supsetneq E_{i+1}$	Other
478,380,414	52,241,971	89,118,791	79,239,945	301,327,548
	521,928,255			
Total: 1,000,308,669				

Table 4.4: Relation between the sets of edges where there have been probe changes.

4.5.3 Remapping observed

We now quantify probe changes and remapping observed across our three snapshots of Sec. 4.4.1. Because identifying remapping requires observing probing changes between two consecutive snapshots, we restrict our analysis to the set of probes with replies from two consecutive snapshots. Of the 6,019,578,262 unique flow IDs that elicited replies in our database (11,689,101,599 replies in total), we find that 3,060,133,214 of them are present in two consecutive snapshots, representing a bit more than half of the flow IDs.

To understand why nearly half of the probe replies do not appear in two consecutive snapshots, we find that for 87% (2,576,532,888) of them the probe ID has in fact been sent only in one snapshot. This is due to the adaptive nature of the D-Miner algorithm resulting in variations of discovery between snapshots presented in Sec. 4.4.4. If, for example, the number of edges discovered for a prefix differs across the three snapshots, the statistical guarantees tell us the number of probes that must be sent will also differ. This results in some probe IDs being sent in only one snapshot. The result is that we are likely underestimating the number of meaningful probe changes. For the remaining 13.0%, the probe was sent in two consecutive snapshots, but did not elicit two replies. This is likely due to normal packet losses in the network and possibly ICMP rate limiting [113].

We start by looking at the number of probe changes. Table 4.3 shows the distribution of the number of probe changes per probe. We see that 28.6% of the probes have at least one probe change. This number seems unusual if we were to consider it all as routing changes.Table 4.4 explains the previous 28.6% fraction by providing the distribution of 1,000,308,669 changes according to the remapping classification. The “Other” column refers to changes with no inclusion relationship but element(s) in common. Notice that the sum of these classifications is slightly smaller than total number of probe changes. The missing probe changes correspond to cases where there was no predecessor for the node corresponding to the IP reply elicited by the probe. This would happen in Figure 4.7

if v_1 was anonymous (a '*') for example.

We see that 52.2% of the probe changes correspond to remapping, which temper the impact of the 28.6% probe changes on routing changes. Finally, for each of the probe changes corresponding to remapping, we performed IP to AS translation on the corresponding IP reply, using August 4, 2019 BGP data from route views [17]. We found that remapping events were spread over 39,455 ASes, showing that this phenomenon is widespread. We conclude this section by noting that all of the changes between snapshots, either due to D-Miner varying the set of probes from one snapshot to another; real dynamics such as routing changes; or remapping due to reboots and/or adding/removing load balanced paths, make any efficiency optimization based on historical discovery hard. It also further corroborates Cunha et al.'s finding that it is difficult to predict path changes [62].

4.6 An updated Internet load balancing survey

Its been eight years since the last published survey of load balancing in the Internet [42]. Whereas this previous study was limited to MDA traceroutes to ∼120k targets, in this section we undertake the task of leveraging D-Miner to perform an exhaustive survey of Internet load balancing on 14,461,947 /24 destination prefixes.

Some things have certainly changed. We now see far larger load balanced topologies, for instance, with thousands of edges, instead of tens. This section updates our understanding of load balancing in the Internet, with some of the more notable results being: 17.9% of load balancing takes place between autonomous systems (i.e., *inter-AS load balancing*); just one autonomous system accounts for the topologies that contain more than 2000 edges; 64.7% of our traces towards all of the /24 prefixes contain at least one branching point (but this might be vantage point dependent); and 1.9% of branching points are per-packet load balancers.

4.6.1 Dataset

From the first snapshot of Sec. 4.4.1 (Aug 6-8, 2019), we extract all of the unique "diamonds" found on the load balanced paths. We adopt the same definition of a diamond as given by Augustin et al. [42]: a *diamond* is "a subgraph delimited by a divergence point followed, two or more hops later, by a convergence point, with the requirement that all flows from source to destination flow through both points." We say that two diamonds are equal if they share the same divergence and convergence points. When the divergence point or the convergence point is a * (i.e., this TTL is "anonymous"), we say that two diamonds are equal if they have identical node sets. In sending probes towards all IPv4 /24 destination prefixes, we extracted 4,029,866 unique diamonds.

4.6.2 Intra- and inter-AS load balancing

Augustin et al. found just one instance of inter-AS load balancing in their 2011 survey [42], whereas we now find it to be a more prevalent practice. We use Oregon Route Views BGP data [17] from 4 August 2019 to map IP addresses of the router interfaces comprising diamonds we discover in the Internet to autonomous systems (ASes). We further map AS numbers to organization names using CAIDA's AS Rank [97].

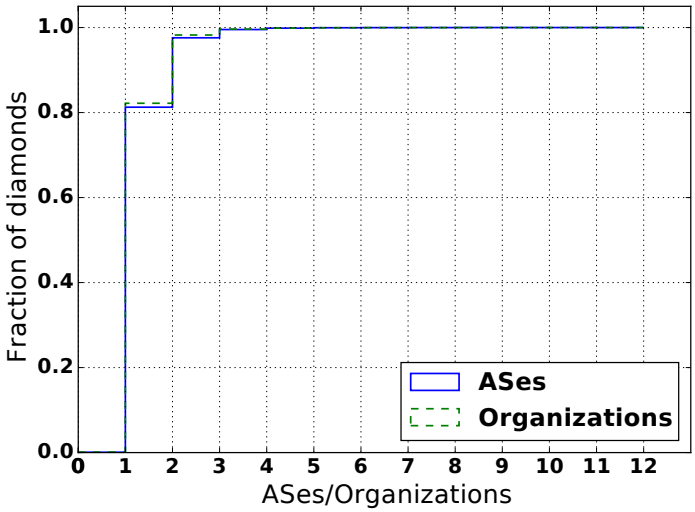
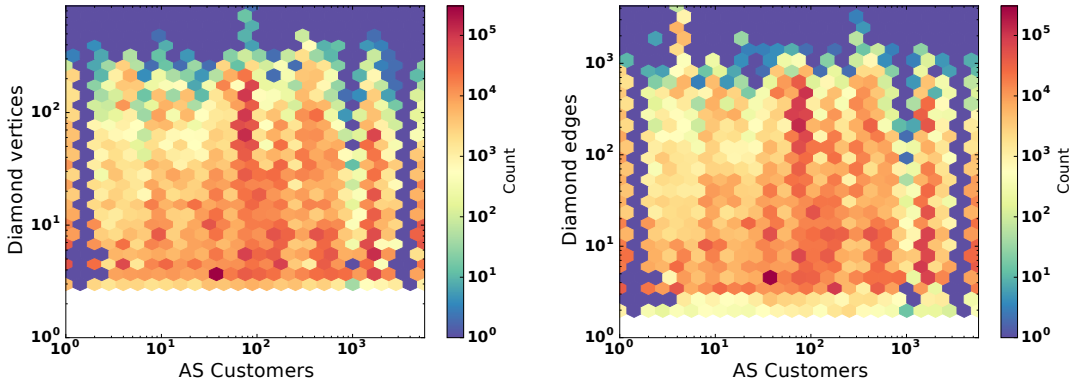


Figure 4.8: Intra- and inter-AS/organization load balancing.



(a) Vertices (b) Edges

Figure 4.9: Joint distribution between the number of customers of ASes and size of the diamonds in these ASes.

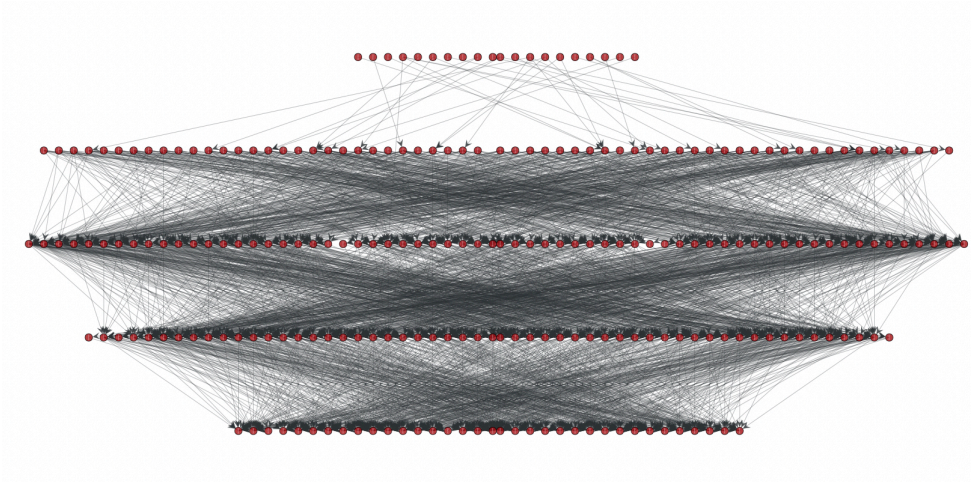


Figure 4.10: Extract of D-Miner trace to an Amazon /24 prefix.

IP address to AS mapping is an outstanding research problem, and correct attribution is known to be difficult [103]. For instance, the IP address of a customer or peer border router is frequently allocated from her provider or peer's address space. We therefore adopt the same methodology of Augustin et al. [42] of not including the diamond's divergence or convergence point when determining the diamond's AS composition. Thus, our estimates of inter-AS load balancing are intended to be conservative.

The CDFs of Figure 4.8 show that, while most load balancing still takes place within a single autonomous system (AS) or organization, a significant portion takes place across two or more of them: 18.7% for ASes and 17.9% for organizations. In one case, we found a single diamond with addresses from 12 ASes (explaining why the CDF continues to 12).

4.6.3 ASes that host load balancers

We next investigate the relationship between the size of an AS (measured as number of customers) and the prevalence of load balancing in that AS. We use AS Rank from CAIDA [97] to perform the AS to customers mapping. When there is more than one AS in the diamond, each AS in the diamond is counted one time. Figure 4.9 shows no clear correlation between the number of customers in an AS and the amount of load balancing we infer, suggesting that load balancing is not limited to large networks, but is a widespread phenomenon.

However, there are some extreme cases involving large networks. We find 74 diamonds of more than 500 nodes each in the network of a French mobile network operator (SFR). Other ASes contain diamonds with $>10^3$ diamond edges, as seen in Figure 4.9b. There are 8,407 diamonds with more than 2,000 edges, of which 8,400 belong to Amazon – likely entry points to their datacenters. The DNS PTR records for the diamond's IP addresses are variations of the address and location, e.g., `ec2-54-178-57-0.ap-northeast-1.compute.amazonaws.com`. These names are characteristic of Amazon's cloud infrastructure. An example of such of a trace is shown in Figure 4.10. This figure shows the last TTLs of D-Miner tracing from a single VP to a single /24 prefix belonging to Amazon. The topology itself strongly resembles current data center design practices that use Clos architectures [132]; we requested validation from Amazon, however, they were unable to provide any corroborating

information due to their internal privacy polices. This example, one of thousands in our data, shows just how complex load balanced topologies can be – complexity that would otherwise be missed without the comprehensive multipath mapping D-Miner provides.

4.7 Conclusion and future work

In this chapter we presented D-Miner, the first Internet-scale system that captures a multipath view of the topology. By combining and adapting state-of-the-art multipath detection and high speed randomized topology discovery techniques, D-Miner permits discovery of the Internet's multipath topology in 2.5 days when probing at 100kpps. This high speed allows us to characterize and quantify dynamic behaviors of the Internet induced by load balancing. Finally, D-Miner enables for the first time an Internet Scale survey of load balancing that shows its widespread prevalence, both in the core and at the edge. We release the D-Miner source code and make our datasets publicly available.

Future work includes running D-Miner with other transport protocols to compare load balancing usage between them at Internet scale, as well as adapting D-Miner to IPv6. Our empirical data suggests that there are a set of load balanced architectures common to different provider types, for instance those deployed in data centers versus mobile operators versus transit providers. We believe there is significant opportunity to develop a taxonomy of these common architectures and classify results accordingly, as well as to identify previously unidentified load balanced architectures that are deployed in the wild. Comprehensive mapping of some of these topologies may require probing both from outside and within the provider's network; we leave an exploration of e.g., internal data center probing to future work.

Finally, in order to provide regular surveys to the community, we wish to deploy D-Miner on more vantage points at high probing speed, create periodic snapshots and perform alias resolution on the resulting discovered topologies.

This chapter ends our contributions to the IP level topology mapping of the Internet. D-Miner showed that it is, as of today, the most complete system to discover the IP level topology from a single server, both in terms of links and nodes.

The next chapter presents a contribution to the router level topology of the Internet, another field of research of the Internet cartography. It proposes an innovative alias resolution technique based on a feature which has always been seen as an impediment to researchers, i.e., ICMP rate limiting.

Chapter 5

Router level topology

The two previous chapters presented our contributions to IP level topology mapping of the Internet. In this chapter, we describe our contributions to another important aspect of the Internet topology, the router level. We introduce Limited Ltd., an innovative technique to perform alias resolution, the process of grouping IP addresses into routers. This work has been published in the proceedings of the *International Conference on Passive and Active Network Measurement 2020*.

5.1 Introduction

Route traces obtained using `traceroute` and similar tools provide the basis for generating maps that reveal the inner structure of the Internet's many autonomously administered networks, but not necessarily at the right level of granularity for certain important tasks. Designing network protocols [131] and understanding fundamental properties of the Internet's topology [73] are best done with router-level maps. Rather than revealing routers, `traceroute` only provides the IP addresses of individual router interfaces. The process of grouping IP addresses into sets that each belong to a common router is called *alias resolution*, and this paper advances the state of the art in alias resolution.

A common approach to alias resolution is to send probe packets to IP addresses, eliciting reply packets that display a feature that is distinctive enough to constitute a signature, allowing replies coming from a common router to be matched. This paper describes a new type of signature based upon a functionality, *ICMP rate limiting*, in which an Internet-connected node (router or end-host) limits the ICMP traffic that it sends or receives within a certain window of time. This new signature enjoys much broader applicability than existing ones for IPv6 alias resolution, thanks to ICMP rate limiting being a required function for IPv6 nodes. The signature also complements IPv4 existing signatures.

Our contributions are: (1) The Limited Ltd. algorithm, a new signature-based alias resolution technique that improves alias resolution coverage by 68.4% on Internet2 for IPv6 and by 40.9% on SWITCH for IPv4 (2) a free, open source, and permissively licensed tool that implements the algorithm.

We evaluate Limited Ltd. by comparing its performance to two state-of-the-art alias resolution tools: Speedtrap [93] for IPv6, and MIDAR [87] for IPv4, using ground truth provided by the Internet2 and SWITCH networks.

The remainder of this chapter is organized as follows: Sec. 5.2 provides technical background and related work for both alias resolution and ICMP rate limiting. Sec. 5.3 describes the Limited Ltd. technique in detail. Sec. 5.4 presents the evaluation. Sec. 5.5 discusses ethical considerations and Sec. 5.6 summarizes our conclusions and points to future work.

5.2 Background and Related Work

Limited Ltd. is the latest in a long line of alias resolution methods stretching back over twenty-plus years. We have explained in chapter 2 the limitations of the current alias resolution techniques, both for IPv4 and IPv6, and motivated the need to create Limited Ltd. .

Regarding ICMP, the Internet Control Message Protocol: its IPv4 and IPv6 variants [110, 59] allow routers or end-hosts to send error and informational messages. The RFC for ICMPv6 [59] cites the "bandwidth and forwarding costs" of originating ICMP messages to motivate the need to limit the rate at which a node originates ICMP messages. It also recommends the use of a token bucket mechanism for rate limiting. It explicitly calls for compatibility with `traceroute` by stating that "Rate-limiting mechanisms that cannot cope with bursty traffic (e.g., traceroute) are not recommended". Furthermore, it states that, in the case of "ICMP messages [being] used to attempt denial-of-service attacks by sending back to back erroneous IP packets", an implementation that correctly deploys the recommended token bucket mechanism "would be protected by the ICMP error rate limiting mechanism". The RFC makes ICMP rate limiting mandatory for all IPv6 nodes. ICMP rate limiting is a supported feature on all modern routers but its implementation may vary by vendor [64, 57, 56, 54, 82, 79, 83, 81] based on ICMP message type and IP version. ICMP rate limiting can be performed on incoming traffic or generated replies. Limited Ltd. makes no distinction between the two. It works whenever multiple interfaces of a router are subject to a common ICMP rate limiting mechanism, i.e., when there is a shared token bucket across multiple interfaces. Vendor documentation [82, 79, 83, 56], indicates that `ping` packets are more likely to trigger shared ICMP rate limiting behavior. We validated this observation in a prior survey and in a lab environment. In particular on Juniper (model J4350, JunOS 8.0R2.8), we observed a shared ICMP rate limiting mechanism for Echo Reply, Destination Unreachable and Time Exceeded packets across all of its interfaces by default. But on Cisco (model 3825, IOS 12.3), we observed that the rates for Time Exceeded and Destination Unreachable packets are limited on individual interfaces by default, and only the rate for Echo Reply packets is shared across different interfaces [55]. Therefore, we adopted the `ping` Echo Request and Echo Reply mechanism in our tool to maximize the chances of encountering shared ICMP rate limits across router interfaces.

A few prior studies have examined ICMP rate limiting behavior in the Internet. Ravaioli et al. [113] identified two types of behavior when triggering ICMP rate limiting of Time Exceeded messages by an interface: on/off and non on/off. Alvarez et al. [36] demonstrated that ICMP Time Exceeded rate limiting is more widespread in IPv6 than in IPv4. Guo and Heidemann [75] later proposed an algorithm, FADER, to detect ICMP Echo Request/Reply rate limiting at very low probing rates, up to 1 packet per second. They found rate limiting at those rates for very few /24 prefixes. Our work is the first

one that exploits the shared nature of ICMP rate limiting across different interfaces of a router as a signature to relate these interfaces for alias resolution.

5.3 Algorithm

The main intuition behind our approach is that two interfaces of a router that implements shared ICMP rate limiting, should exhibit a similar loss pattern if they are both probed by ICMP packets at a cumulative rate that triggers rate limiting. The key challenges are to efficiently trigger rate limiting and reliably associate aliases based on the similarity of their loss patterns despite the noise due to independent losses of probes and replies.

Pseudo code 1 describes how Limited Ltd. divides a set of input IP addresses into subsets that should each be an alias set. It proceeds iteratively, taking the following steps in each iteration: First, a random IP address from the input set is selected as a *seed*, with all remaining members of the input set being *candidate* aliases for the seed. The seed is probed at incrementally higher rates until the rate r_s that induces ICMP rate limiting is identified (`find_rate`()). Then, the seed is probed at that rate of r_s while all of the candidates interfaces are simultaneously probed at low rates. All probing takes place from a single vantage point. Loss traces for reply packets from the seed and each of the candidate interfaces are gathered. It is very challenging to infer that two interfaces are aliases by directly correlating their loss traces. Instead, the algorithm extracts a set of features from each loss trace and collectively uses these as the signatures of the corresponding interfaces(`signatures`()). Using a classification technique (`classify`()), the algorithm examines whether the signatures of candidate and seed are sufficiently similar to classify them as aliases, in which case the candidate is added to an alias set (A_s). Each identified alias set is refined through further testing in order to reduce the chance of false positives (`refine`()). Finally, the alias set is removed from the input set, and iterations continue until the input set is empty. The remainder of this section further details these steps.

5.3.1 Triggering ICMP rate limiting

The goal of `find_rate`(s) is to efficiently determine r_s, the probing rate that triggers ICMP rate limiting at the router to which seed s belongs. It proceeds by probing the seed with ICMP Echo Request probes across multiple rounds, increasing the probing rate with each round until the loss rate of observed ICMP Echo Replies enters a target range. The target loss range should be sufficiently large to minimize the effect of random independent losses and also relatively small to minimize the load on the router. To satisfy these two opposing conditions, we empirically set the range at 5 to 10%. The probing rate remains constant during each round. The rate is low (64 pps) for the first round, and exponentially increases in consecutive rounds until the loss rate falls within (or exceeds) the target range.[1] If the observed loss rate is within the target range, the probing is concluded and the last rate is reported as r_s. But if the loss rate is higher than the target range, up to eight additional rounds are launched in a binary search between the last two rates.

[1] We have explicitly verified that the actual probing rate is not limited by the network card or other factors.

Algorithm 1 `Limited Ltd.`

Input
S: a set of IP addresses
Output
A: a set of alias sets
$K \leftarrow$ `controls`(S) : set of controls
$A \leftarrow \emptyset$
while $S \neq \emptyset$ **do**
choose at random a seed $s \in S$
$C_s \leftarrow S \setminus \{s\}$: candidate aliases for s
$r_s \leftarrow$ `find_rate`(s) : rate limiting rate for s
$\Sigma_s \leftarrow$ `signatures`(s, r_s, C_s, K) : set of signatures
$A_s \leftarrow \{s\}$: alias set for s
for $c \in C_s$ **do**: for each candidate c
$\sigma_{s,c} \in \Sigma_s$ is the pairwise signature for s and c
if `classify`$(\sigma_{s,c})$ == `true` **then**
s and c are aliases
$A_s \leftarrow A_s \cup \{c\}$: add c to the alias set
$A_s \leftarrow$ `refine`(A_s) : try to reduce false positives
$A \leftarrow A \cup \{A_s\}$: add the new alias set to A
$S \leftarrow S \setminus A_s$: remove the aliases of s from S
return A

If the loss rate still does not fall within the target range, the probing rate that generates the loss rate closest to the range is chosen. If the target loss range is not reached as the probing reaches a maximum rate (32,768 pps), the probing process ends without any conclusion. The duration of each round of probing should be sufficiently long to reliably capture the loss rate while it should also be limited to control the overhead of probing. We experimentally set the duration of each round of probing to 5 seconds, followed by a period, of equal length, of no probing. The right plot of Fig. 5.1 presents the CDF of the number of probing rounds to trigger the target loss rate for thousands of IPv4 and IPv6 interfaces (using our dataset from Sec. 5.3.3). We observe that for 90% of IPv4 or IPv6 interfaces, the ICMP rate limiting is triggered in less than 8 rounds of probing. The left plot of Fig. 5.1 shows the CDF of the probing rate that triggered the target loss rate (i.e., the inferred rate for triggering the ICMP rate limiting) across the same IPv4 and IPv6 interfaces. This figure indicates that for 70% (80%) of IPv6 (IPv4) interfaces, ICMP rate limiting is triggered at less than 2k pps. This result confirms that our selected min and max probing rate covers a proper probing range for more than 99% of interfaces. We note that the binary search process failed to reach the target loss rate for fewer than 1% of the interfaces. All the parameters of our probing strategy are empirically determined. Section 5.5 elaborates on the ethical considerations associated with the probing scheme.

5.3.2 Generating interface signatures

A signature based on the loss traces of individual interfaces is obtained by probing the seed interface at its target rate (r_s) while simultaneously probing each candidate interface at the low rate of R_c pps. Probing a large number of candidate interfaces in each round

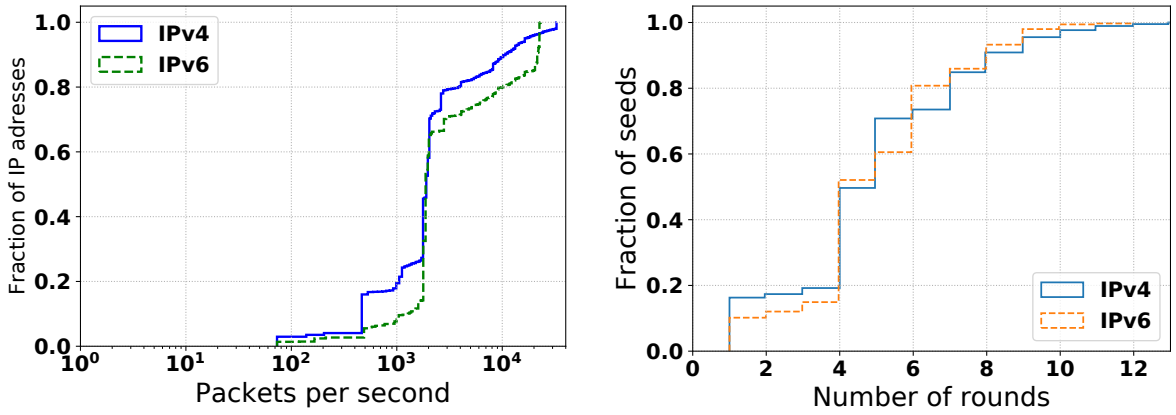


Figure 5.1: CDF of the probing rate r_s (left) and the number of probing rounds (right) to trigger ICMP rate limiting for 2,277 IPv4 and 1,099 IPv6 addresses.

may lead to a better efficiency, but the aggregate probing rate should remain low so that it does not independently trigger ICMP rate limiting even if all those candidates are in fact aliases. To address these two constraints, we set the number of candidate interfaces that are considered in each round to 50 and R_c to 10 pps. In an unlikely scenario that all of these 50 candidate interfaces are aliases, this strategy leads to a 500 pps probing rate for the corresponding router that does not trigger ICMP rate limiting in 90% of routers, as we showed in the left plot of Fig. 5.1.[2]

Control Interface.

In order to distinguish the observed losses in the loss traces for the target interfaces (i.e., seed s and individual candidate c) that are not related to ICMP rate limiting, we also consider another interface along the route to each target interface and concurrently probe them at a low rate (10 pps). These interfaces are called the *controls*, κ_s and κ_c. The control κ_i for target interface i is identified by conducting a Paris Traceroute [38] towards i and selecting the last responsive IP address prior to i.[3] The loss rate for κ_i also forms part of i's signature. In practice, the *controls* are identified at the beginning of the Limited Ltd. procedure by conducting route traces to all IP addresses in the input set S. This corresponds to `controls`() and K is the resulting set of controls.

Inferring Alias Pairs.

The above probing strategy produces a separate loss trace for each interface. We have found that when losses occur simultaneously at pairs of alias interfaces, they can do so in multiple ways, as the five examples in Fig. 5.4 illustrate. The black and white strokes in each trace correspond respectively to received and lost ICMP Echo Replies, and their varied patterns defy attempts to find simple correlations. We therefore use a machine learning classifier to identify pairs of aliases. It is based on the following features

[2]The largest reported alias set by MIDAR and Speedtrap has 43 interfaces. Therefore, the likelihood of observing 50 candidate interfaces that are all aliases is low.

[3]Limited Ltd. maintains the flow ID necessary to reach κ_s in subsequent probing of s and κ_s.

	Seed s	Candidate c	Control κ_s	Control κ_c
Loss rate	x	x	x	x
Change point	x	x		
`gap` → `gap` transition probability	x	x		
`burst` → `burst` transition probability	x	x		
Pearson correlation coefficient	x			

Table 5.1: Selected features for a Signature.

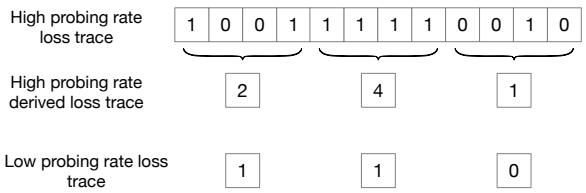


Figure 5.2: Mapping between loss traces with different length.

extracted from loss traces that, intuitively, we believe capture the temporal pattern of the losses in each trace. (See also Table 5.1.)

1. Loss rate: This is simply the number of losses in the trace divided by the total number of probes in the trace.

2. Change point detection: This is the point in a time series (such as our loss traces) when the probability distribution of a time series changes [37]. We adopt a method based on the variation of the mean and the variance [88].

3. Transition probabilities: These are obtained by using each loss trace to train a Gilbert-Elliot two-state Markov model, in which losses occur in the `burst` state and no losses occur in the `gap` state. The P(`gap` → `gap`) and P(`burst` → `burst`) transition probabilities are sufficient to fully describe the model since other two probabilities can be easily calculated from these. For example, P(`gap` → `burst`) = 1 - P(`gap` → `gap`).

4. Correlation coefficient: The Pearson correlation coefficient between the two loss traces is used as a measure of similarity between them. Calculating this coefficient requires both time series to have the same number of values but our loss traces do not meet this condition since we use a higher probing rate for the seed. To address this issue, we condition the seed's loss trace to align it with the loss trace of other interfaces as shown in Fig. 5.2. In this example, the length of the loss trace of the seed is four times longer than the ones from the other interfaces. We consider groups of four consecutive bits in the seed loss trace and convert it to the sum of the 1's. The resulting loss trace has a lower rate and can be directly correlated with other loss traces.

5.3.3 Classifying the signatures

We use the *random forest* classifier from the `scikit-learn` Python machine learning library [109]. If it identifies two interfaces as aliases based on their signatures, `classify`() returns `true`; otherwise, `false`. There are several challenges to building such a classifier: (1) it must learn from training data that represents the diversity of possible loss traces

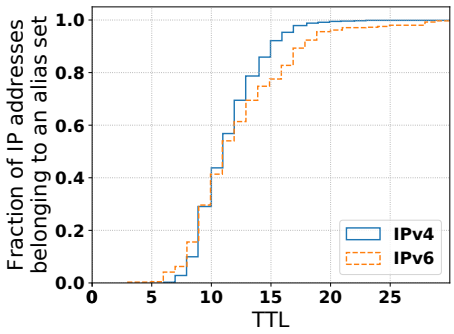


Figure 5.3: CDF of the TTL distance from the Limited Ltd. vantage point of the IP addresses belonging to an alias set in our training data.

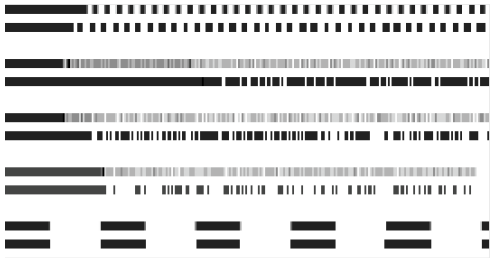


Figure 5.4: Raw times series of loss traces of pairs of aliases.

generated by pairs of aliases; (2) it should be able to distinguish between losses triggered by ICMP rate limiting and unrelated losses; (3) it should have a high precision, so that Limited Ltd. minimizes false positives; and (4) if the training data come from other alias resolution techniques, such as MIDAR and Speedtrap, it must be able to generalize to pairs that they cannot find. We tackled these challenges as follows.

Training and testing data.

We have access to ground truth router-level topology for two networks, Internet2 and SWITCH, but these do not suffice to capture the richness of router behaviors in the Internet as a whole. We therefore randomly selected routable IPv4 and IPv6 prefixes from the RIPE registry [24], and conducted multipath Paris Traceroute [129] from PlanetLab Europe [19] nodes towards the first address in each prefix. This procedure yielded 25,172 IPv4 addresses in 1,671 autonomous systems (ASes) and 18,346 IPv6 addresses in 1,759 ASes from 6,246 and 4,185 route traces, respectively. We use MIDAR and Speedtrap to identify IPv4 and IPv6 alias sets, respectively, since both tools are known to have low false positive rates. Pairs of interfaces from these sets are used as labeled as `true`. For the `false` labels, we take the conservative approach of selecting pairs of IP addresses that are more than 6 hops from each other in a given route trace. The 6 hop value is empirically set, as 99.9% of the alias pairs identified by MIDAR and Speedtrap are fewer than 6 hops apart. This labeling process identified 70,992 unique IPv4 and 7,000 unique IPv6 addresses. 15,747 of IPv4 and 1,616 IPv6 addresses are labeled as aliases forming 2,277 IPv4 and 1,099 IPv6 alias sets, respectively. Fig. 5.3 shows the CDF of hop count distance between our vantage point and selected IP addresses and indicates that these targets are 7-17 hops away from the vantage point. For each alias set, one address is chosen at random to play the role of the seed s, and the candidate set is composed of all of the other aliases in the set that are rounded up with some randomly selected non-aliases to make a C_s of size between 2 (minimum one alias and one non-alias) and 50 (our cap for the number of addresses to be simultaneously probed at a low rate). The high rate r_s at which to probe the seed is found through `find_rate`(s), and the signatures are generated through `signatures`(s, r_s, C_s, K).

	IPv4			IPv6		
	Precision	Recall	F1 score	Precision	Recall	F1 score
Random forest	0.990	0.499	0.652	0.992	0.647	0.782
Multilayer perceptron	0.993	0.431	0.591	0.978	0.641	0.769
KNN	0.952	0.638	0.764	0.970	0.622	0.756
SVM	0.986	0.478	0.642	0.988	0.599	0.743

Table 5.2: Classifier performance on our test set averaged over ten training/testings.

	Gini index	
Feature	IPv4	IPv6
loss rate for the candidate c	0.169	0.192
`burst` $\rightarrow$ `burst` transition probability for the candidate c	0.113	0.125
`burst` $\rightarrow$ `burst` transition probability for the seed s	0.101	0.121
Pearson correlation coefficient	0.091	0.109
loss rate for κ_c, the control of the candidate c	0.077	0.104

Table 5.3: The five most important features of our random forest classifiers.

Note that while our classifier is trained on alias sets identified by alias resolution techniques with known limitations, it is nonetheless able to identify new alias sets. We argue that this is because the training set is sufficiently rich due to its size and random selection of interfaces, providing considerable diversity and heterogeneity of loss traces across aliases. Our evaluation in Sec. 5.4 confirms this observation and confirms the ability of our technique to generalize patterns in the training dataset, i.e., the fourth aforementioned challenge.

Choice of classifier.

We compared the performance of four classifiers that `scikit-learn` library offers, namely random forest, multilayer perceptron, k-nearest neighbors (KNN), and support vector machines (SVM). To this end, we evenly divided our dataset into a training and a test set, and compared these classifiers based on their precision, recall, and F1 score for both IPv4 and IPv6 datasets. Since `true` labels are only provided from aliases identified by MIDAR and Speedtrap, the recall values correspond to the portion of pairs of aliases in our training set that are detectable by both MIDAR and Limited Ltd. (IPv4) or by both Speedtrap and Limited Ltd. (IPv6). Table 5.2 presents the averaged result of this comparison after performing 10 randomized splits of the training and test sets. All classifiers exhibit relatively good performance. We have decided to use the random forest classifier, which is composed of 500 trees, as it has the highest precision for both IPv4 and IPv6, and the best F1 score for the IPv6 dataset.

Finally, Table 5.3 shows the five most important features of our random forest classifiers based on the Gini index [48] that describes the weight of individual features in the classifier's decision. This table reveals a few important points. First, no single feature dominates the classifier's decision, particularly for IPv6. This confirms the complexity of the patterns for relating loss traces of aliases, as they cannot be accurately learned by a small number of features or simple threshold-based rules. Second, this table also illustrates that most of our engineered features are indeed very important in distinguishing loss traces of aliases. Third, the use of κ_c as one of the main features suggests that the classifier distinguishes losses related to rate limiting from other losses.

5.3.4 Refining the alias set

Independent network loss could accidentally result in classifying unrelated interfaces as aliases, i.e., generating false positives. To reduce the chance of this, Limited Ltd. incorporates a refinement step, `refine(`A_s`)`, that involves repeating `signature()` and `classify()` on the previously-identified alias set A_s. If a candidate c fails to be (re)classified as an alias of the seed s, it is removed from the alias set. This step is repeated until the alias set remains unchanged over two iterations. Sec. 5.4 evaluates the resulting reduction of false positives.

5.4 Evaluation

We evaluate Limited Ltd. with regards to its ability (i) to identify alias pairs that state-of-the-art techniques, namely MIDAR and Speedtrap, are unable to identify, and (ii) to maintain a low rate of false positives.

Dataset.

We evaluate Limited Ltd. on ground truth data from the Internet2 and SWITCH networks. For Internet2, router configuration files were obtained on 10 April, with measurements conducted on 11 and 12 April 2019. There were 44 files, each corresponding to a single router. All are Juniper routers. The files concern 985 IPv4 and 803 IPv6 addresses/interfaces, from which we removed 436 IPv4 addresses and 435 IPv6 addresses that did not respond to any probes sent by either MIDAR, Speedtrap, or Limited Ltd. . The resulting dataset consists of 6,577 IPv4 and 2,556 IPv6 alias pairs. For SWITCH, a single file was obtained on 3 May, with measurements conducted 3-5 May 2019. The file identified 173 Cisco routers running either IOS or IOS-XR. From the 1,073 IPv4 and 706 IPv6 addresses listed in the file, we removed 121 IPv4 and 29 IPv6 unresponsive addresses. The resulting dataset consists of 4,912 IPv4 and 2,641 IPv6 alias pairs.

Reducing false positives.

We computed the distribution of number of rounds for `refine`() to finalize the alias set for each seed in our dataset: For 79% (98%) of all seeds, `refine`() takes 2 (3) more rounds. Note that the minimum of two rounds is required by design (Sec. 5.3.4) This basically implies that `refine`() only changed the alias set for 20% of the seeds in a single round.

Results.

Table 5.4 presents the precision and recall of MIDAR, Speedtrap, Limited Ltd. , and the union of both tools on IPv4 and IPv6 ground truth data from the Internet2 and SWITCH networks. Note that it is possible for recall from the union of both tools to be greater than the sum of recall values for individual tools, as we observe in the SWITCH results. This arises from the transitive closure of alias sets identified from the two tools that leads to the detection of additional alias pairs. The main findings of Table 5.4 can be summarized as follows:

		IPv4			IPv6		
		MIDAR	ltd ltd	MIDAR ∪ ltd ltd	Speedtrap	ltd ltd	Speedtrap ∪ ltd ltd
Internet2	Precision	1.000	1.000	1.000	N/A	1.000	1.000
	Recall	0.673	0.800	0.868	N/A	0.684	0.684
SWITCH	Precision	1.000	1.000	1.000	1.000	1.000	1.000
	Recall	0.090	0.499	0.599	0.384	0.385	0.772

Table 5.4: Evaluation on ground truth networks.

1. Limited Ltd. exhibits a high precision in identifying both IPv4 and IPv6 alias pairs from both networks with zero false positives.

2. Limited Ltd. can effectively discover IPv6 aliases that state-of-the-art Speedtrap is unable to find. In the Internet2 network that uses Juniper routers, Limited Ltd. was able to identify 68.4% of the IPv6 alias pairs while Speedtrap was unable to identify any. In the SWITCH network that deploys Cisco routers, Limited Ltd. and Speedtrap show comparable performance by identifying 38.5% and 38.4% of the IPv6 alias pairs, respectively. The results were complementary, with the two tools together identifying 77.2% of the IPv6 alias pairs, a small boost beyond simple addition of the two results coming from the transitive closure of the alias sets found by each tool.

3. Limited Ltd. can discover IPv4 aliases that state-of-the-art MIDAR is unable to find. In the Internet2 network, Limited Ltd. identifies 80.0% while MIDAR detects 67.3% of aliases. In the SWITCH networks, Limited Ltd. identified 49.9% while MIDAR detects only 9.0% of all aliases.

A couple of detailed observations follow. We conducted follow up analysis on the behavior of Speedtrap and MIDAR to ensure proper assessment of these tools. First, we examined Speedtrap's logs to diagnose Speedtrap's inability to detect any IPv6 aliases for Internet2. We noticed that every fragmentation identifier time series that Speedtrap seeks to use as a signature, was either labeled as random or unresponsive. This was not surprising, as prior work on Speedtrap [93] also reported that this technique does not apply to the Juniper routers that primarily comprise Internet2. Second, we explored MIDAR's logs to investigate the cause of its low recall for SWITCH. We learned that only one third of the IPv4 addresses in this network have monotonically increasing IP IDs.

Limitations and future work.

Because ICMP rate limiting could be triggered at thousands of packets per second, Limited Ltd. requires the sending of many more packets than other state-of-the-art alias resolution techniques. The maximum observed probing rate during the experiments for this paper was 34,000 pps from a single vantage point during a 5-second round. On Internet2 (SWITCH), MIDAR and Speedtrap sent 164.5k (106k) and 4k (12.7k) probe packets while Limited Ltd. sent about 4,8M (12.7M) packets. In future work, we plan to explore ways to reduce the overhead of probing and make Limited Ltd. more scalable.

5.5 Ethical Considerations

Limited Ltd. works by triggering limits in routers that are there for protective reasons. This raises ethical concerns, which we discuss below. To evaluate the impact of Limited Ltd. , we have taken two steps: experiments in a lab environment (Sec. 5.5.1), and feedback from operators (Sec. 5.5.2).

Precautions taken We take two precautions, that we understand to be community best practice: We sent all probing traffic from IP addresses that were clearly associated via WHOIS with their host locations, either at our institution or others hosting PlanetLab Europe nodes. We have also set up a web server on the probing machines with a contact email, so that any network operators could opt out from our experiment. We received no notice whatsoever from network operators expressing concern about our measurements. Though this is a positive sign, it could be that there are impacts that were not noticed, or that the concerns did not reach us. We therefore pushed our examination further, as detailed in the following sections.

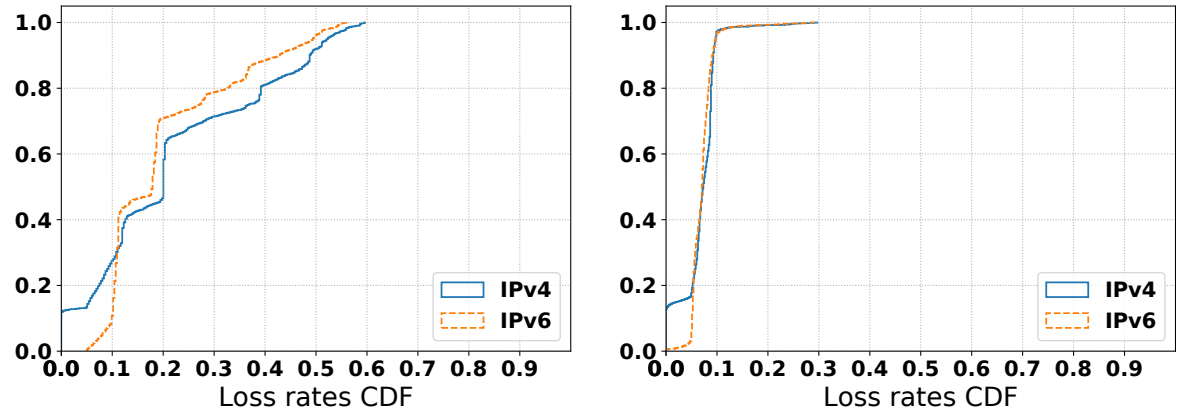
5.5.1 Lab experiments

We have run experiments in a lab environment on conservatively chosen hardware (over 10 years old) to show that Limited Ltd. has a controlled impact. Our findings are that: (1) routers being probed with Echo Requests by the tool remain reachable to others via `ping` with a high probability; and (2) Router CPUs show a manageable overhead at the highest probing rate, leading us to believe that our measurements are unlikely to impact the control and data planes. (3) Both Limited Ltd. and existing measurement techniques impact troubleshooting efforts (e.g., `ping`, `traceroute`). Limited Ltd. does not stand out in terms of impact compared with other accepted techniques. Next paragraph details the experiments which support these conclusions.

Impact on other measurements.

Limited Ltd. 's `find_rate`() aims to find an ICMP Echo Request probing rate that produces an Echo Reply trace with a loss rate in the $[0.05, 0.10]$ range. While it is searching for this rate, it can induce a loss rate above 0.10. If it does so, it proceeds to a binary search to find a lower probing rate for which traces falls within the desired range. Fig. 5.5 shows that loss rates can go as high as 0.60.

The impact on reachability for the IP addresses of that node is that there is a worst case 0.60 probability that a single `ping` packet to such an address will not receive a response if it arrives at the node during the five seconds of highest rate probing time. Most pings occur in series of packets, so the worst case probabilities are 0.36 for two `ping` packets being lost, 0.22 for three, 0.13 for four, 0.08 for five, and 0.05 for six. These are worst case probabilities for the five seconds at highest loss rate. Average reachability failure probabilities are 0.22 for one `ping` packet, 0.05 for two, 0.01 for three, and so on, while a node is being probed at its highest rate. To judge whether such a level of interference with other measurements is exceptional, we compare it to the impact of the state-of-the-art MIDAR tool. MIDAR has a phase during which it elicits three series of 30 responses each, using different methods for each series: TCP SYN packets, to elicit TCP



(a) For `find_rate`()

(b) For `signatures`()

Figure 5.5: Maximum loss rates

```
traceroute to 83.99.240.90 (83.99.240.90), 30 hops max, 60 byte packets
 1  132.227.123.1  0.350 ms  0.308 ms  0.363 ms
 2  134.157.167.125  0.342 ms  0.435 ms  0.415 ms
 3  134.157.254.124  0.495 ms  0.468 ms  0.444 ms
 4  195.221.127.181  0.359 ms  0.350 ms  0.327 ms
 5  193.51.181.102  1.326 ms  1.325 ms  1.303 ms
 6  193.51.180.108  1.395 ms  1.290 ms 193.51.177.117  1.239 ms
 7  83.97.89.9  1.029 ms  1.011 ms  0.989 ms
 8  * * *
 9  62.40.98.36  8.879 ms  8.866 ms  8.951 ms
10  * * *
11  87.245.233.222  38.814 ms  38.846 ms  38.902 ms
12  87.245.242.26  39.142 ms 87.245.242.29  38.878 ms 87.245.242.26  39.012 ms
13  85.254.1.16  39.105 ms 85.254.1.128  39.029 ms 85.254.1.16  39.236 ms
14  82.193.72.232  39.287 ms  39.548 ms 82.193.72.130  39.519 ms
15  83.99.240.90  45.451 ms * *
```

(a) Before and after MIDAR run

```
traceroute to 83.99.240.90 (83.99.240.90), 30 hops max, 60 byte packets
 1  132.227.123.1  0.436 ms  0.341 ms  0.343 ms
 2  * * *
 3  134.157.254.124  0.421 ms  0.457 ms  0.434 ms
 4  195.221.127.181  0.358 ms  0.349 ms  0.319 ms
 5  193.51.181.102  1.209 ms  1.196 ms  1.200 ms
 6  193.51.177.117  0.860 ms 193.51.180.108  0.948 ms 193.51.177.117  0.896 ms
 7  83.97.89.9  0.975 ms  0.973 ms  0.950 ms
 8  * * *
 9  62.40.98.36  8.565 ms  8.607 ms  8.544 ms
10  * * *
11  87.245.233.222  38.801 ms  38.768 ms  38.748 ms
12  87.245.242.29  38.945 ms 87.245.242.26  39.009 ms  38.988 ms
13  85.254.1.16  39.028 ms  39.065 ms  39.070 ms
14  82.193.72.232  39.238 ms  39.426 ms 82.193.72.130  39.400 ms
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * 83.99.240.90  45.314 ms *
```

(b) While MIDAR is running

Figure 5.6: Example erroneous `traceroute` result

RST or TCP SYN-ACK responses; UDP packets to a high port number, to elicit ICMP Destination Unreachable responses; and ICMP Echo Request packets, to elicit ICMP Echo Reply responses [87]. The probing rate is very low compared to Limited Ltd. : a mere 100 packets per second across multiple addresses. This is not a concern for the TCP and ICMP probing. However, the UDP probing taps into an ICMP rate limiting mechanism that tends to be much less robust than the typical ICMP Echo Reply mechanism on some routers. ICMP Destination Unreachable messages are often rate limited at 2 packets per second, which is $1/500^{\text{th}}$ the typical rate at which ICMP Echo Reply messages are rate limited. (For example, the default rate at which Cisco routers limit ICMP Destination Unreachable messages is 1 every 500 ms.)

We found that, when an IP address is a `traceroute` destination, MIDAR can completely block ICMP Destination Unreachable messages coming from that destination. Fig. 5.6 illustrates the impact. The figure shows two `traceroute` results, the top one from before or after MIDAR being run, and the bottom one during MIDAR probing. During the MIDAR run, we see that `traceroute` receives no responses while it is probing hop 15, where the destination is in fact to be found. The normal functioning of `traceroute` is to continue probing at higher and higher hop counts. Only a few seconds later, when `traceroute` is sending probes to hop 20, does it start to receive ICMP Destination Un-

reachable messages from the destination. The result is an erroneous `traceroute`, indicating that the destination is five hops further away than it actually is. We observed this erroneous `traceroute` effect on 2,196 IP addresses out of a dataset of 10,000 IPv4 addresses collected from across the Internet. For both Limited Ltd. and MIDAR, transient interference with other measurements can be observed for the few seconds during which an IP address is being probed. Our conclusion is not that the diminution in `ping` reachability induced by Limited Ltd. is necessarily anodyne. Care should be taken to circumscribe this effect. But we observe that it does not stand out in terms of its impact on other measurements.

CPU usage

We now examine the CPU overhead generated by Limited Ltd. , and its potential impact on the forwarding plane and other features involving the CPU. We have run an experiment in a local network with our own Cisco (model 3825, IOS 12.3) and Juniper (model J4350, JunOS 8.0R2.8) routers. The experiment consists in measuring three metrics while `find_rate`() routine of Limited Ltd. , which has the highest probing rate, is running. We measured: (1) The CPU usage of the router, (2) the throughput of a TCP connection between the two end hosts, and (3) the rate of BGP updates. ICMP rate limiting is configured on both our Juniper and Cisco routers with an access list [80, 55], limiting the ICMP input bandwidth destined to the router to 1,000 packets per second, which is the default configuration on Juniper routers.

TCP throughput was unaffected, at an average of 537 Mbps and BGP updates remained constant at 10 per second. CPU usage was at 5% for Cisco and 15% for Juniper when Limited Ltd. was not probing. During the probing, the maximum overhead was triggered for both at a maximum probing rate of 2,048 packets per second, with a peak at 10% for Cisco and 40% for Juniper during 5 seconds. Our conclusion is that there is an impact of high probing rates on CPU, but we do not witness a disruptive impact on either the data plane (TCP throughput) or the control plane (BGP update rate).

5.5.2 Real-world operator feedback

In addition to lab experiments, we conducted joint experiments with SURFNET and SWITCH to evaluate the potential impact of Limited Ltd. . The experiment consisted in running Limited Ltd. on their routers while they were monitoring the CPU usage. Each run lasted about 1 minute. For SURFNET, we ran Limited Ltd. on two Juniper routers: an MX240 and an MX204. The operator observed a 4% and 2% CPU overhead. The operator also told us that the CPU overhead was observed on the MPC (line modules) CPU and not the central routing engine CPU. For SWITCH, we ran Limited Ltd. on three Cisco routers: an NCS 55A1, an ASR 9001, and an ASR-920-24SZ-M. On the two first routers, the operator told us that there was no observable change in CPU utilization. On the third router, which has a lower CPU capacity than the two others, the operator observed a CPU overhead up to 29%. These results confirm our belief that Limited Ltd. is unlikely to impact the control and data planes.

5.6 Conclusion

This chapter presented Limited Ltd., a new high-precision alias resolution technique for both IPv4 and IPv6 networks that leverages the ICMP rate limiting feature of individual routers. We have shown that ICMP rate limiting can generate loss traces that can be used to reliably identify aliases from other interfaces. Limited Ltd. enables IPv6 alias resolution on networks composed of Juniper routers that the state-of-the-art Speedtrap technique is not able to identify. However, Limited Ltd. has limitations such as its important overhead in terms of the number of packets sent.

This concludes our contribution on alias resolution, which showed to be a new promising technique that enlarges the spectrum of routers that are eligible for alias resolution. The next chapter concludes this dissertation.

Chapter 6

Conclusion

This thesis presented our three major contributions to the Internet cartography: Multilevel MDA-Lite Paris Traceroute (MMLPT), Diamond-Miner, and Limited Ltd.. We conclude this thesis by giving a summary of our contributions (Sec. 6.1), and putting our thesis into perspective (Sec. 6.2).

6.1 Summary of contributions

Each piece of work provides new insights into the Internet cartography. In Multilevel MDA-Lite Paris Traceroute, we provided a new tool for researchers and operators, which provides the most complete IP and router level view compared with other traceroute tools. MMLPT revealed that load balanced topologies have become a lot more complex than 10 years ago.

Then, based on these results, we designed and implemented Diamond-Miner, a system that allows for the first time to perform an Internet scale snapshot of load balanced topologies within 2.5 days, while providing the same level of statistical guarantees as the classic MDA. It confirmed our first results in MMLPT, revealing outstanding topologies with thousands of edges previously unknown. We also gave a new characterization of Internet multipath dynamics. This study showed that the multipath view is essential to a better understanding of routing changes.

Finally, we made an innovative contribution to alias resolution, by leveraging a new signature based on ICMP rate limiting. Limited Ltd. allows the community to perform alias resolution on Juniper routers for IPv6 for the first time, while renewing our decreasing capacity to perform alias resolution on IPv4 routers with IP ID techniques.

All of these new tools and systems are available to the scientific community; we make all our code publicly available.

6.2 Perspectives

This section concludes our dissertation on “Improved algorithms for capturing Internet maps”. It presents where the field stands after this thesis and the future directions that we consider promising.

6.2.1 IP level topology

Prior to this thesis, the community had a partial knowledge of multipath complexity; no results had revealed multipath topologies with more than 16 interfaces at a single hop. Our surveys in chapter 3 and 4 revealed multipath topologies of one order of magnitude more in nodes (92 interfaces at a single hop), and two orders of magnitude more in links (> 5k). Furthermore, prior to our work, the community was unable to perform an exhaustive snapshot of the IP level topology of the Internet inclusive of multipaths in a reasonable time. This is now possible with the Diamond-Miner system. Regarding the metrics defined in the introduction, it is a major improvement in the completeness of IP level topology.

There is still room for improvement in IP level topology capture though. Indeed, a lot of vantage points currently performing IP level topology mapping [1] or able to do so [19, 23] do not have the capacity to run the Diamond-Miner system at its full power. The next challenge is therefore to coordinate heterogeneous vantage points in terms of probing and computation capacity to obtain maximum unique discovery from the entire system.

Another aspect that we have not sought to tackle in this thesis is to reveal MPLS tunnels. It is an important part of Internet topology, and there is an entire literature on discovering MPLS tunnels [66, 123, 124, 99]. To the best of our knowledge, there is no Internet scale system that reveals MPLS tunnels. Diamond-Miner concept of decoupling probing and computation could be adapted to reveal MPLS tunnels at Internet scale.

Last but not least, with the growing adoption of IPv6, a very straightforward contribution based on our work is to translate Multilevel MDA-Lite Paris Traceroute and Diamond-Miner to IPv6.

6.2.2 Router level topology

Prior to this thesis, IP ID (IPv4) and fragmentation Identifier (IPv6) techniques (that rely on the same algorithmic concepts) had the monopoly in alias resolution, with all the limitations that this implies (see chapter 2). Our work proposes fresh ideas in the field, such as leveraging other types of signature, such as ICMP rate limiting, and using machine learning to transform a signal coming from routers into alias labels.

With the decreasing applicability of the IP ID technique in IPv4, we hope that future research will leverage those different ideas to build new alias resolution techniques.

Another aspect of the router level topology that is of primary interest is to reduce the time needed to perform alias resolution. The only production system run by CAIDA that performs alias resolution [3], takes several weeks to perform alias resolution on the whole Internet. This process takes weeks especially because alias resolution techniques make no use of metadata that could help to precluster IP addresses into smaller candidate sets that would not overlap any alias set. MIDAR, the alias resolution technique run by CAIDA, takes a giant set of 2 million IP addresses in input. With preclustering, alias resolution could be run in parallel and see its time to completion reduced drastically.

As a result, the current state is that even if we are now capable of performing exhaustive snapshots of the IP level topology in a couple of days, we are limited by alias resolution running time to obtain router level topology.

6.2.3 Dynamics

The last perspective that our work has opened is the possibility to better capture the dynamics of the Internet topology at Internet scale. Prior to our work, there had been no characterization of IP level dynamics at Internet scale. Indeed, the multipath view was until now not available to the community at Internet scale. And, as we highlighted in chapter 4, the multipath view is essential to this study. Although our characterization had the merit to be the first tackling Internet scale dynamics of the IP level topology, it has been performed on a limited number of snapshots, during a period of a week. A long term study would be worth investigating.

Then, these IP level dynamics could be translated into router level dynamics to better infer the root causes of the dynamics, e.g., is it a routing change, multipath dynamic, or an outage?

With the public access to Diamond-Miner code and collected snapshots, we hope that researchers will be able to better capture and apprehend the dynamics of the Internet.

Last word

I believe that my thesis has developed "Improved algorithms for capturing Internet maps". In the ocean of complexity that the Internet represents, I wish that my work has set some milestones on the long path of a thorough understanding of the Internet.

Bibliography

[1] Ark. https://www.caida.org/projects/ark/locations.

[2] C++ Mersenne Twister. http://www.cplusplus.com/reference/random/mt19937/.

[3] Caida itdk. http://www.caida.org/data/internet-topology-data-kit/index.xml.

[4] CEF polarization. https://www.cisco.com/c/en/us/support/docs/ip/express-forwarding-cef/116376-technote-cef-00.html.

[5] Clickhouse - open source distributed column-oriented DBMS. https://clickhouse.yandex.

[6] Configuring a load-balancing scheme. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipswitch_cef/configuration/xe-3s/isw-cef-xe-3s-book/isw-cef-load-balancing.pdf.

[7] D-miner topology snapshots. https://gitlab.planet-lab.eu/cartography/.

[8] EdgeNet. http://edge-net.org/.

[9] GNS3 emulator. https://www.gns3.com.

[10] Hohio validation. https://www.caida.org/~mjl/rnc/.

[11] How Does Unequal Cost Path Load Balancing (Variance) Work in IGRP and EIGRP? https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13677-19.html.

[12] kapar. https://www.caida.org/tools/measurement/kapar/.

[13] libnetfilter-queue library, netfilter.org project. http://fr.netfilter.org.

[14] libparistraceroute library. https://github.com/libparistraceroute/libparistraceroute.

[15] libtins packet crafting and sniffing library. http://libtins.github.io.

[16] M-Lab. https://www.measurementlab.net.

[17] Oregon route views. http://www.routeviews.org/.

[18] Per-flow and per-packet load balancing. https://support.huawei.com/enterprise/en/doc/EDOC1100055041/ebc8ad42.

[19] PlanetLab Europe. https://www.planet-lab.eu.

[20] Private communication with CAIDA.

[21] Resilient hashing on LAGs and ECMP groups. https://www.juniper.net/documentation/en_US/junos/topics/topic-map/switches-interface-resilient-hashing.html.

[22] Rfc 792. https://tools.ietf.org/html/rfc792.

[23] RIPE Atlas. https://atlas.ripe.net.

[24] RIPE Registry. https://www.ripe.net/publications/docs/ripe-508.

[25] Traceroute for linux. http://traceroute.sourceforge.net.

[26] Troubleshooting load balancing over parallel links using cisco express forwarding. https://www.cisco.com/c/en/us/support/docs/ip/express-forwarding-cef/18285-loadbal-cef.html#internalmech.

[27] Understanding the algorithm used to hash LAG bundle and egress next-hop ECMP traffic (QFX 10002 and QFX 10008 switches). https://www.juniper.net/documentation/en_US/junos/topics/concept/interfaces-hashing-lag-ecmp-understanding-\10002-10008.html.

[28] Understanding the algorithm used to load balance traffic on MX series routers. https://www.juniper.net/documentation/en_US/junos/topics/concept/hash-computation-mpcs-understanding.html.

[29] Understanding the hash algorithm. https://support.huawei.com/enterprise/en/doc/EDOC1100086965.

[30] USC/ISI ANT Datasets. https://ant.isi.edu/datasets/all.html.

[31] Website for topology validation . http://heartbeat.planet-lab.eu.

[32] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. *Journal of the ACM (JACM)*, 56(4):1–28, 2009.

[33] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large european ixp. *ACM SIGCOMM Computer Communication Review*, 42(4):163–174, 2012.

[34] D. Alderson, L. Li, W. Willinger, and J. C. Doyle. Understanding internet topology: principles, models, and validation. *IEEE/ACM Transactions on networking*, 13(6):1205–1218, 2005.

[35] R. Almeida, O. Fonseca, E. Fazzion, D. Guedes, W. Meira, and Í. Cunha. A characterization of load balancing on the ipv6 internet. In *Proc. PAM '17*, 2017.

[36] P. Alvarez, F. Oprea, and J. Rule. Rate-limiting of IPv6 Traceroutes is Widespread: Measurements and Mitigations. In *Proc. IETF 99*, 2017.

[37] S. Aminikhanghahi and D. J. Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[38] 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. IMC '06*, 2006.

[39] B. Augustin, T. Friedman, and R. Teixeira. Exhaustive Path Tracing With Paris Traceroute. In *Proc. CoNEXT '07*, 2006.

[40] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *Proc. IMC '07*, 2007.

[41] B. Augustin, T. Friedman, and R. Teixeira. Multipath tracing with paris traceroute. In *Proc. E2EMON '07*, 2007.

[42] B. Augustin, T. Friedman, and R. Teixeira. Measuring multipath routing in the internet. *IEEE/ACM Transactions on Networking (TON)*, 19(3):830–840, 2011.

[43] A. Barbeiro. Visualizing multipath networks with Dublin Traceroute. Presentation at the MOCA Italian hacker camp, Aug 2016.

[44] S. M. Bellovin. A technique for counting NATted hosts. In *Proc. IMW '02*, 2002.

[45] A. Bender, R. Sherwood, and N. Spring. Fixing Ally's Growing Pains with Velocity Modeling. In *Proc. IMC '08*, 2008.

[46] R. Beverly. Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery. In *Proc. IMC '16*, 2016.

[47] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer. In the IP of the beholder: Strategies for active IPv6 topology discovery. In *Proc. ACM IMC '18*, 2018.

[48] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[49] CAIDA. The CAIDA ark IPv4 Internet topology data kits dataset, 2019. `http://www.impactcybertrust.org`.

[50] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications magazine*, 35(6):160–163, 1997.

[51] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for Internet load balancing. In *Proc. INFOCOM '00*, pages 332–341, 2000.

[52] Y. Chang and P. Boothe. A better schema for paris-traceroute. Presentation at AIMS '18 CAIDA Workshop, Mar 2018.

[53] W. Chen, Y. Huang, B. F. Ribeiro, K. Suh, H. Zhang, E. d. S. e Silva, J. Kurose, and D. Towsley. Exploiting the ipid field to infer network path and end-system characteristics. In *Proc. PAM '05*, 2005.

[54] Cisco. Cisco IOS quality of service solutions configuration guide, release 12.2SR; chapter: Policing and shaping overview. https://www.cisco.com/c/en/us/td/docs/ios/qos/configuration/guide/12_2sr/qos_12_2sr_book/polcing_shping_oview.html.

[55] Cisco. Configure commonly used IP ACLs. https://www.cisco.com/c/en/us/support/docs/ip/access-lists/26448-ACLsamples.html.

[56] Cisco. Control plane policing implementation best practices. https://www.cisco.com/c/en/us/about/security-center/copp-best-practices.html#7.

[57] Cisco. IPv6 ICMP rate limiting. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_basic/configuration/xe-3s/ip6b-xe-3s-book/ip6-icmp-rate-lmt-xe.pdf.

[58] k. claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet Mapping: from Art to Science. In *IEEE DHS Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, 2009.

[59] A. Conta and M. Gupta. RFC 4443, Internet Control Message Protocol (ICMPv6) for the Internet Protocol version 6 (IPv6) specification, IETF. 2006.

[60] M. Cotton, L. Vegoda, R. Bonica, and B. Haberman. Special-purpose IP address registries. RFC 6890 (Best Current Practice), Apr. 2013.

[61] Í. Cunha, R. Teixeira, and C. Diot. Measuring and characterizing end-to-end route dynamics in the presence of load balancing. In *International Conference on Passive and Active Network Measurement*, pages 235–244. Springer, 2011.

[62] Í. Cunha, R. Teixeira, D. Veitch, and C. Diot. Dtrack: a system to predict and track internet path changes. *IEEE/ACM Transactions on Networking (TON)*, 22(4):1025–1038, 2014.

[63] A. Dainotti, C. Squarcella, E. Aben, K. C. Claffy, M. Chiesa, M. Russo, and A. Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 1–18, 2011.

[64] R. A. Deal. Cisco router firewall security: DoS protection. http://www.ciscopress.com/articles/article.asp?p=345618&seqNum=5.

[65] B. Donnet and T. Friedman. Internet topology discovery: a survey. *IEEE Communications Surveys & Tutorials*, 9(4):56–69, 2007.

[66] B. Donnet, M. Luckie, P. Mérindol, and J.-J. Pansiot. Revealing mpls tunnels obscured from traceroute. *ACM SIGCOMM Computer Communication Review*, 42(2):87–93, 2012.

[67] C. Dovrolis, K. Gummadi, A. Kuzmanovic, and S. D. Meinrath. Measurement lab: Overview and an invitation to the research community. *ACM SIGCOMM Comput. Commun. Rev.*, 40(3):53–56, 2010.

[68] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall. Detecting intentional packet drops on the Internet via TCP/IP side channels. In *Proc. PAM*, 2014.

[69] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999.

[70] M. Ferrante and M. Saltalamacchia. The coupon collector's problem. *MATerials MATemàtics*, 2014(2):1–35, May 2014.

[71] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6):733–745, 2001.

[72] R. Govindan and H. Tangmunarunkit. Heuristics For Internet Map Discovery. In *Proc. INFOCOM '00*, 2000.

[73] M. H. Gunes and K. Sarac. Importance of IP Alias Resolution in Sampling Internet Topologies. In *Proc. GI '07*, 2007.

[74] M. H. Gunes and K. Sarac. Resolving IP Aliases in Building Traceroute-Based Internet Maps. *IEEE/ACM Transactions on Networking*, 17(6):1738–1751, 2009.

[75] H. Guo and J. Heidemann. Detecting ICMP Rate Limiting in the Internet. In *Proc. PAM '18*, 2018.

[76] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an autonomous system (as), 1996.

[77] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley. ns-3 project goals. In *Proc. WNS2 '06*, 2006.

[78] V. Jacobson. 4BSD routing diagnostic tool available for ftp. Email 8812201313.AA03127@helios.ee.lbl.gov to the IETF and end2end-interest e-mail lists, 1988.

[79] Juniper. Default ICMP rate limit on the system for host inbound connections. `https://kb.juniper.net/InfoCenter/index?page=content&id=KB28184&cat=SRX_SERIES&actp=LIST`.

[80] Juniper. IPv6 multicast routing on E series broadband services routers, release 15.1; access-list. `https://www.juniper.net/documentation/en_US/junose15.1/topics/reference/command-summary/access-list.html`.

[81] Juniper. Policer implementation overview. `https://www.juniper.net/documentation/en_US/junos/topics/concept/policer-mx-m120-m320-implementation-overview.html`.

[82] Juniper. System management and monitoring feature guide for switches; internet-options (ICMPv4). `https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration-statement/icmpv4-rate-limit-edit-system.html`.

[83] Juniper. System management and monitoring feature guide for switches; internet-options (ICMPv6). `https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration-statement/icmpv6-rate-limit-edit-system.html`.

[84] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards ip geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84. ACM, 2006.

[85] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. E. Anderson, and A. Krishnamurthy. Reverse traceroute. In *Proc. NSDI '10*, 2010.

[86] K. Keys. Internet-Scale IP Alias Resolution Techniques. *ACM SIGCOMM Computer Communication Review*, 40(1):50–55, 2010.

[87] K. Keys, Y. Hyun, M. Luckie, and K. Claffy. Internet-Scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM Transactions on Networking*, 21(2):383–399, 2013.

[88] R. Killick and I. A. Eckley. changepoint: An R package for changepoint analysis. *Journal of Statistical Software*, 58(3):1–19, 2014.

[89] S. Kim and K. Harfoush. Efficient Estimation of More Detailed Internet IP Maps. In *Proc. ICC '07*, 2007.

[90] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 190–201. ACM, 2009.

[91] M. Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *Proc. IMC '10*, 2010.

[92] M. Luckie and R. Beverly. The impact of router outages on the as-level internet. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 488–501. ACM, 2017.

[93] M. Luckie, R. Beverly, W. Brinkmeyer, et al. Speedtrap: internet-Scale IPv6 Alias Resolution. In *Proc. IMC '13*, 2013.

[94] M. Luckie, A. Dhamdhere, K. Claffy, and D. Murrell. Measured impact of crooked traceroute. *ACM SIGCOMM Computer Communication Review*, 41(1):14–21, 2011.

[95] M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark, et al. bdrmap: inference of borders between ip networks. In *Proc. ACM IMC '16*, 2016.

[96] M. Luckie, B. Huffaker, and k. claffy. Learning regexes to extract router names from hostnames. In *Proceedings of the Internet Measurement Conference*, pages 337–350, 2019.

[97] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, et al. As relationships, customer cones, and validation. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 243–256. ACM, 2013.

[98] M. Luckie, Y. Hyun, and B. Huffaker. Traceroute probe method and forward IP path inference. In *Proc. ACM IMC '08*, 2008.

[99] J.-R. Luttringer, Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet. Let there be light: Revealing hidden mpls tunnels with tnt. *IEEE Transactions on Network and Service Management*, 2019.

[100] P. Marchetta, A. Montieri, V. Persico, A. Pescapé, Í. Cunha, and E. Katz-Bassett. How and how much traceroute confuses our understanding of network paths. In *Proc. LANMAN '16*, 2016.

[101] P. Marchetta, V. Persico, and A. Pescapè. Pythia: Yet Another Active Probing Technique for Alias Resolution. In *Proc. CoNEXT '13*, 2013.

[102] A. Marder. Apple: Alias pruning by path length estimation. In *International Conference on Passive and Active Network Measurement*, pages 249–263. Springer, 2020.

[103] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, J. M. Smith, et al. Pushing the boundaries with bdrmapIT: Mapping router ownership at Internet scale. In *Proc. ACM IMC '18*, 2018.

[104] R. Motamedi, B. Yeganeh, B. Chandrasekaran, R. Rejaie, B. M. Maggs, and W. Willinger. On mapping the interconnections in todayâĂŹs internet. *IEEE/ACM Transactions on Networking*, 27(5):2056–2070, 2019.

[105] D. J. Newman. The double Dixie Cup problem. *The American Mathematical Monthly*, 67(1):58–61, 1960.

[106] R. Padmanabhan, Z. Li, D. Levin, and N. Spring. UAv6: AliasRresolution in IPv6 Using Unused Addresses. In *Proc. PAM '15*, 2015.

[107] J.-J. Pansiot and D. Grad. On Routes and Multicast Trees in the Internet. *ACM SIGCOMM Computer Communication Review*, 28(1):41–50, 1998.

[108] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM transactions on networking*, 7(3):277–292, 1999.

[109] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[110] J. Postel et al. Rfc 792: Internet control message protocol. *InterNet Network Working Group*, 1981.

[111] S. Qian, Y. Wang, and K. Xu. Utilizing Destination Options Header to Resolve IPv6 Alias Resolution. In *Proc. GLOBECOM '10*, 2010.

[112] S. Qian, M. Xu, Z. Qiao, and K. Xu. Route Positional Method for IPv6 Alias Resolution. In *Proc. ICCCN '10*, 2010.

[113] R. Ravaioli, G. Urvoy-Keller, and C. Barakat. Characterizing ICMP Rate Limitation on Routers. In *Proc. ICC '15*, 2015.

[114] Y. Rekhter, T. Li, S. Hares, et al. RFC 4271: A border gateway protocol 4 (bgp-4), 1994.

[115] RIPE NCC Staff. RIPE Atlas. *Internet Protocol Journal*, 18(3), 2015.

[116] J. P. Rohrer. Operationalizing yarrp: High-speed active network topology mapping from aws. `https://www.caida.org/workshops/kismet/2002/slides/kismet2002_jrohrer.pdf`.

[117] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications*, 29(9):1810–1821, 2011.

[118] J. Sherry, E. Katz-Bassett, M. Pimenova, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy. Resolving ip aliases with prespecified timestamps. In *Proc. IMC '10*, 2010.

[119] R. Sherwood, A. Bender, and N. Spring. Discarte: a Disjunctive Internet Cartographer. *ACM SIGCOMM Computer Communication Review*, 38(4):303–314, 2008.

[120] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[121] Stephen D. Strowes. Visibility of IPv4 and IPv6 Prefix Lengths in 2019, 2019. `https://labs.ripe.net/Members/stephen_strowes/visibility-of-prefix-lengths-in-ipv4-and-ipv6`.

[122] J. P. Sterbenz, E. K. Çetinkaya, M. A. Hameed, A. Jabbar, S. Qian, and J. P. Rohrer. Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation. *Telecommunication systems*, 52(2):705–736, 2013.

[123] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet. MPLS under the microscope: Revealing actual transit path diversity. In *Proc. ACM IMC '15*, 2015.

[124] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet. Through the wormhole: tracking invisible mpls tunnels. In *Proceedings of the 2017 Internet Measurement Conference*, pages 29–42, 2017.

[125] Y. Vanaubel, J.-J. Pansiot, P. Mérindol, and B. Donnet. Network Fingerprinting: TTL-Based Router Signatures. In *Proc. IMC '13*, 2013.

[126] D. Veitch, B. Augustin, R. Teixeira, and T. Friedman. Failure control in multipath route tracing. In *IEEE INFOCOM 2009*, pages 1395–1403. IEEE, 2009.

[127] K. Vermeulen, B. Ljuma, V. Addanki, M. Gouel, O. Fourmaux, T. Friedman, and R. Rejaie. Alias resolution based on icmp rate limiting. In *International Conference on Passive and Active Network Measurement*, pages 231–248. Springer, 2020.

[128] K. Vermeulen, J. P. Rohrer, R. Beverly, O. Fourmaux, and T. Friedman. Diamond-miner: Comprehensive discovery of the internet's topology diamonds. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 479–493, 2020.

[129] K. Vermeulen, S. D. Strowes, O. Fourmaux, and T. Friedman. Multilevel MDA-Lite Paris Traceroute. In *Proc. IMC '18*, 2018.

[130] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, and R. Teixeira. Detection, understanding, and prevention of traceroute measurement artifacts. *Computer Networks*, 52(5):998–1018, Apr. 2008.

[131] W. Willinger, D. Alderson, and J. C. Doyle. Mathematics and the Internet: A Source of Enormous Confusion and Great Potential. *Notices of the American Mathematical Society*, 56(5):586–599, 2009.

[132] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan. Understanding Lifecycle Management Complexity of Datacenter Topologies. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019.

[133] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: Detecting IP prefix hijacking on my own. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 327–338, New York, NY, USA, 2008. ACM.

[134] S. Zhou and R. J. Mondragón. Accurately modeling the internet topology. *Physical Review E*, 70(6):066108, 2004.