Diamond-Miner: Comprehensive Discovery of the Internet's Topology Diamonds

Kevin Vermeulen, Justin P Rohrer, Robert Beverly, Olivier Fourmaux, Timur Friedman

Diamond-Miner: Comprehensive Discovery of the Internet's Topology Diamonds

Kevin Vermeulen
Sorbonne Université

Justin P. Rohrer
Naval Postgraduate School

Robert Beverly
Naval Postgraduate School

Olivier Fourmaux
Sorbonne Université

Timur Friedman
Sorbonne Université

Abstract

Despite the well-known existence of load-balanced forwarding paths in the Internet, current active topology Internet-wide mapping efforts are multipath agnostic – largely because of the probing volume and time required for existing multipath discovery techniques. This paper introduces D-Miner, a system that marries previous work on high-speed probing with multipath discovery to make Internet-wide topology mapping, inclusive of load-balanced paths, feasible. We deploy D-Miner and collect multiple IPv4 interface-level topology snapshots, where we find >64% more edges, and significantly more complex topologies relative to existing systems. We further scrutinize topological changes between snapshots and attribute forwarding differences not to routing or policy changes, but to load balancer "remapping" events. We precisely categorize remapping events and find that they are a much more frequent contributor of path changes than previously recognized. By making D-Miner and our collected Internet-wide topologies publicly available, we hope to help facilitate better understanding of the Internet's true structure and resilience.

1 Introduction

An important component of today's Internet is multipath routing [18, 29, 45], where traffic to a destination network is *load-balanced* to support higher capacities and provide redundancy. Prior work developed active measurement techniques to discover multipath routing [17], while showing that multipath topologies can be quite complex [45]. However, high probing loads and runtime have been impediments to their widespread uptake. As a result, today's IP and router topology snapshots, e.g., [21, 41], 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 [15, 24, 46]. Obtaining not just accurate, but complete topologies is crucial to understanding the network's resilience to outages and attacks [35, 43], as well as aiding in the conception of applications ranging from content distribution to network security [48]. Further, these topologies play a vital supporting role in other measurement inferences, for instance determining AS relationships, mapping inter-domain congestion [32], and geolocation [31] 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 [19] and ii) multipath detection algorithms [17]. 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 (§3). Having implemented D-Miner, we evaluate its performance and overhead (§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 (§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 (§6). Our contributions thus include:

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 [4].

2 Background and related work

We first review different types of load balancing commonly found in the Internet. Then, we provide an overview of the Multipath Detection Algorithm (MDA) [44], the current state-of-the-art technique for actively discovering load balancing, and Yarrp, a method for high-speed topology discovery. We finish this section by describing other related work.

2.1 Load balancing

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 [3, 6, 12]: deterministic and non-deterministic. When a packet arrives on a router configured with deterministic load balancing (i.e., per-flow load balancing), and multiple equal-cost routing paths are available to the packet's destination, the router chooses a path by computing a hash over the packet's header fields [23]. This set of fields used to compute the hash is called the flow identifier, and typically includes either the source and destination addresses (per-destination) or the source and destination addresses and ports (per-flow). 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. Herein, we use the terms "flow identifier" and "flow" interchangeably.

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.

Our Internet scale survey in §6 confirms two previous results of Augustin et al. [18] and Vermeulen et al. [45]: (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.

2.2 MDA

The Traceroute tool [30] sends probe packets to find forwarding paths. It exploits the IPv4 time-to-live (TTL) header field to induce routers along a forwarding path to send ICMP error messages, thereby revealing the router's interface addresses. The original Traceroute design did not foresee the later emergence of load-balanced paths in the Internet, and it gives incomplete and incorrect results in the face of load balancing [16]. Paris Traceroute [16] was developed specifically to accurately reveal a path through a per-flow load-balanced network. Paris Traceroute ensures that all probe packets, across different TTLs, have consistent flow identifiers, thereby ensuring that all measurement packets take the same path in a load-balanced network. However, in its basic implementation, Paris Traceroute reveals only a single path to the destination.

The Multipath Detection Algorithm (MDA) stochastically varies Paris Traceroute's flow identifiers in an attempt to enumerate all paths to a destination. For a given vertex with k known outgoing load-balanced edges, the number of probes with randomly selected flow IDs needed to verify that it has no more than k edges is denoted n_k, and is termed a "stopping point" [44]. 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.

Unfortunately, the stateful nature of the MDA and its reliance on establishing confidence in the behavior of each potential branching point along the path in a sequential manner are hinderences to its use for Internet-wide topology studies.

Note, the MDA technique has previously been validated [18, 44]. From this perspective, if D-Miner achieves the same level of statistical guarantees that the MDA provides, it validates D-Miner as well. §4 shows that D-Miner fulfills this condition.

2.3 Yarrp

Yarrp [19, 20] introduced the notion of high-speed topology probing via stateless operation, and random permutation of targets and TTLs. Whereas previous route tracing techniques, e.g., [34], iteratively probe TTLs toward each destination, Yarrp randomizes its probing and decouples probing from topology reconstruction. This randomization avoids overloading particular paths or routers, thereby permitting higher probing rates. Further, Yarrp encodes all of the necessary state, e.g., originating TTL and time, into probes such that the quoted replies permit state to be reconstituted. In this fashion, Yarrp demonstrated the ability to perform Internet-wide route tracing at more than 100k pps.

2.4 Other related work

For more than two decades now, Internet mapping has been an active area of research. It allows researchers to better un-

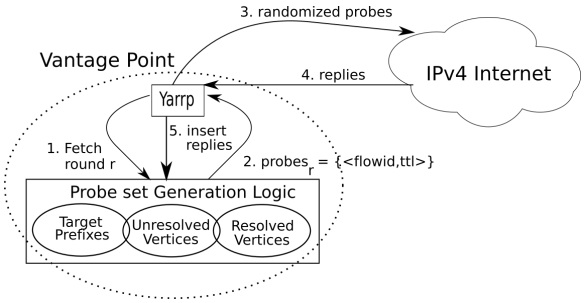


Figure 1: D-Miner high-level conceptual overview

derstand the structure of the Internet [22, 28] and to design better protocols [46]. Today, one can either obtain an Internet snapshot of the whole Internet without load-balanced paths, or a reduced snapshot of the Internet with load-balanced paths. Mapping systems such as CAIDA's Ark [24] perform continuous surveys from hundreds of vantage points by launching Paris Traceroute measurements to one random address in every globally advertised /24 prefix. A complete set of measurements towards all of the /24 prefixes is called a cycle. Because the address in the /24 is randomly chosen, aggregating results from multiple cycles reveal some load balancing. 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. Yarrp [19], described in §2.3, performs Internet scale surveys at high speed, but is also not capable of revealing the load-balanced paths. D-Miner enables researchers, for the first time, to obtain snapshots 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 [18]. 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 [45]. In both of those studies, the set of destination prefixes was a subset of the entire Internet, with, respectively, $\sim$120k and $\sim$350k targets. In contrast our survey contains traces to $\sim$14.4M /24 destination prefixes.

D-Miner allows us to provide new insights into Internet dynamics induced by load balancing. In §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 [39] 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 [26], 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 [27], 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.

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 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 §3.2.

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 [25], 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 §2.2, 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 [42].

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 §2.3, 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.

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 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 [2]. 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 §3.3 and the analyses of §4, §5, and §6 tractable.

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.

3.3.1 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 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 [18, 27].

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 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 §2.2.

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:

$$\forall v, E[X_v] = D_h(v)N >= n_{k_v} \tag{1}$$

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

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

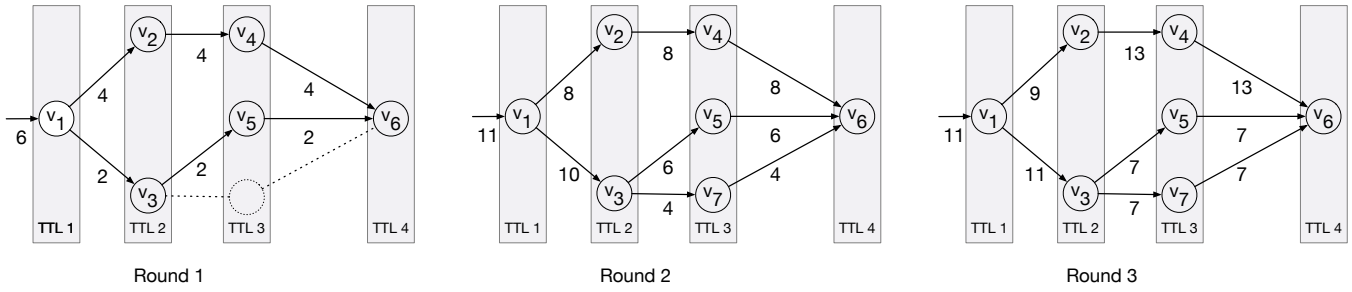


Figure 2: Example topology resolved by three rounds of D-Miner probing. Link annotations represent number of probes expiring at ingress interface of subsequent node.

We have:

$$\forall v, E[X_v] = D_h(v) \max_{v' \in R_h} \frac{n_{k_{v'}}}{D_h(v')} >= D_h(v) \frac{n_{k_v}}{D_h(v)} = n_{k_v} \quad (2)$$

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. 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 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.

3.3.2 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 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.

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 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.

3.5 Per-Packet load balancing

As described in §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 [18].

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 §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 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 (§4.4.1), 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 §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. §4.3 evaluates D-Miner's running time. First, however, §4.1 describes the datasets we collected, and shows that 10 rounds suffice to reach statistical guarantees for 99% of the /24 destination prefixes.

4.1 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).

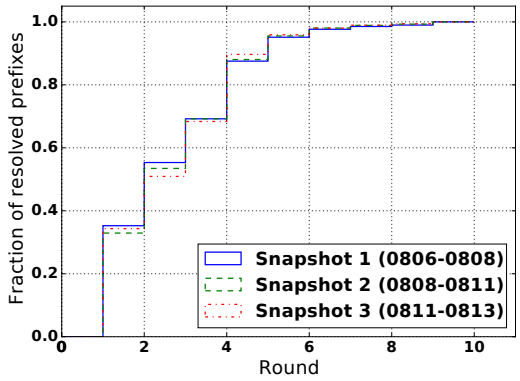


Figure 3: CDF of /24 IPv4 prefixes meeting their statistical guarantees, over three 10-round snapshots (dates in parens)

Table 1: Probing Overhead of D-Miner versus MDA.

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
Last round (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**

In addition, for the topology discovery section, we have deployed D-Miner on six PlanetLab Europe (PLE) [7] 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 [36]. 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 3 illustrates how 10 rounds of probing suffices to achieve statistical guarantees 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%.

4.2 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

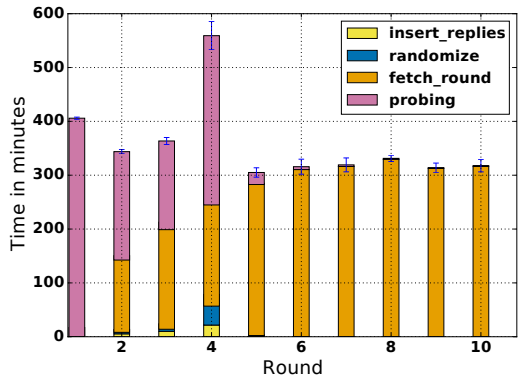


Figure 4: Time spent in each step of D-Miner for each round, averaged across three snapshots.

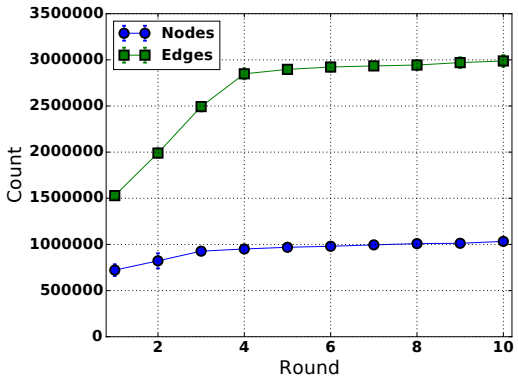


Figure 5: Per-round node / edge discovery for three snapshots.

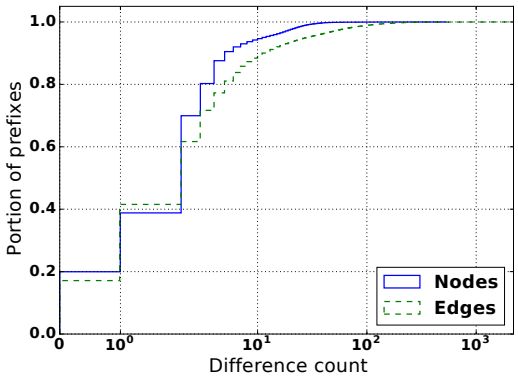


Figure 6: CDF of the minimum difference of number of nodes and edges per prefix.

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 in the last round than required to reach the statistical guarantees. Table 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 last round overhead, we simulate for each of the destination prefixes a run of the MDA with the same flow identifiers and compute the actual probe at which it stops due to having reached the statistical guarantees.

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 shows the stacked error bars 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) and the fraction of resolved prefixes (Figure 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 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.

[1] Represented as a * in traditional traceroute output [9]

Table 2: Topology discovery

	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			

Figure 5 shows the cumulative discovery, with error bars, 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 2 gives the comparative topology results. The "Agg." column shows the aggregated results over the three snapshots. The two last lines of Table 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 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 §5.2). Indeed, the difference is not attributable to high probing rate induced loss, as Table 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 [36] 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.

4.4.1 Probes sent

In total, we compute that Ark sent 5,935,460,660 probes. From Table 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.

4.5 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 belong-

ing 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 [14] 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.

5 Forwarding path dynamics

With D-Miner, we reveal aspects of Internet dynamics that were under estimated [26] 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 [26,38]. 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.

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* [18] 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 §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*.

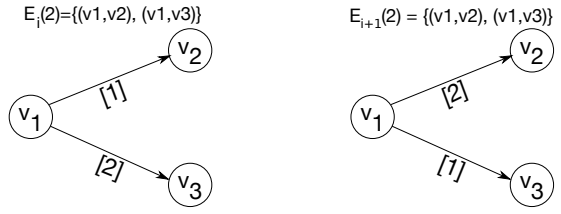
5.2 Load balancer remapping

Production systems are more nuanced in their behavior relative to the overview of load balancing in §2.1. 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* [1, 11, 13], which occurs when load balancers are chained and apply the same hashing algorithm, potentially causing load imbalance.

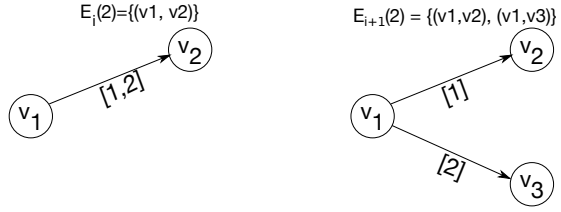
We experimented with Cisco routers running IOS 12 in a lab environment and confirmed that this seed is configurable. If the Cisco router reboots and has no saved seed, it generates a new random seed. For Juniper and Huawei, documentation tells us that they also use a seed [11, 13].

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 [8, 10]. 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*.

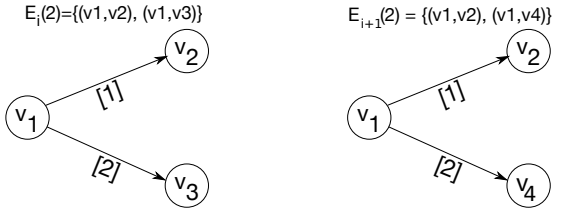
We attempt to identify in our dataset probe changes that correspond specifically to load balancer remapping. Our conservative rule of thumb is to say that if we observe a meaningful change for probe $(X,h+1)$, but at least one of the



(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 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.

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 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 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 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 7c shows an example of no inclusion relation where there is both a new and deleted edge due to remapping.

Table 3: Changes per (flow ID, TTL) probe on three snapshots.

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

Table 4: Relation between the sets of edges where there have been probe changes.

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

5.3 Remapping observed

We now quantify probe changes and remapping observed across our three snapshots of §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 §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 [40].

We start by looking at the number of probe changes. Table 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 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 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 [5]. 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 [27].

6 An updated Internet load balancing survey

Its been eight years since the last published survey of load balancing in the Internet [18]. 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.

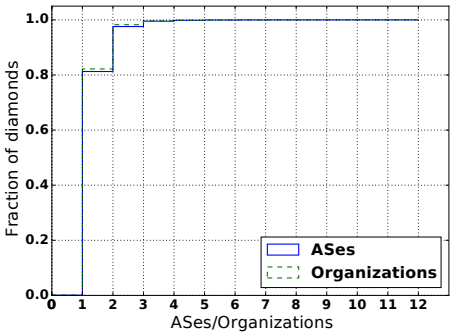


Figure 8: Intra- and inter-AS/organization load balancing.

6.1 Dataset

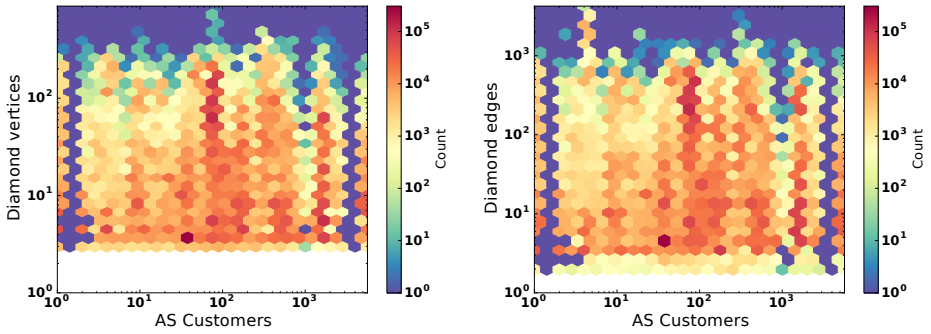
From the first snapshot of §4.1 (Aug 6-8, 2019), we extract all of the unique "diamonds" found on the load balanced paths. We adopt the same definition of a diamond as given by Augustin et al. [18]: a *diamond* is "a subgraph delimited by a divergence point followed, two or more hops later, by a convergence 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.

6.2 Intra- and inter-AS load balancing

Augustin et al. found just one instance of inter-AS load balancing in their 2011 survey [18], whereas we now find it to be a more prevalent practice. We use Oregon Route Views BGP data [5] 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 [33].

IP address to AS mapping is an outstanding research problem, and correct attribution is known to be difficult [37]. 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. [18] 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 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).



(a) Vertices (b) Edges

Figure 9: Joint distribution between the number of customers of ASes and size of the diamonds in these ASes.

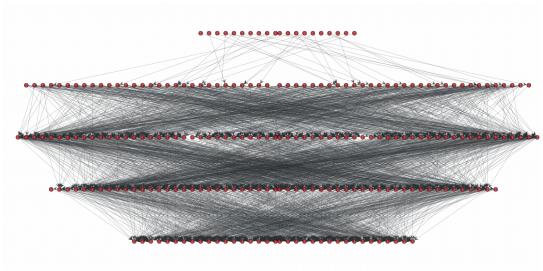


Figure 10: Extract of D-Miner trace to an Amazon /24 prefix.

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 [33] 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 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 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 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 [47]; 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.

7 Conclusion and future work

In this work we present 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.

Our hope is that D-Miner and our data will facilitate better understanding of the Internet's true structure, properties, and resilience. 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.

Acknowledgments

We thank: Ophelie Walrick, for suggesting the name Diamond-Miner; Matthieu Gouel, for the website for topology validation; network administrators at Sorbonne Université and the Renater NREN, for assistance in running our measurements; Romain Fontugne, from IIJ, Niels den Otter, from SURFNet, Bernd Spiess, from IP-IT, Olva Kvittem, from UNINETT, and Marc Ammann, from SUNRISE, for their ground truth validation; the anonymous reviewers from NSDI, and our shepherd, Ben Zhao, for their careful reading and suggestions. Robert Beverly and Justin P. Rohrer are associated with Naval Postgraduate School, department of Computer Science. Kevin Vermeulen, Olivier Fourmaux, and Timur Friedman are associated with Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, F-75005 Paris, France. Kevin Vermeulen and Timur Friedman are associated with the Laboratory of Information, Networking and Communication Sciences, LINCS, F-75013 Paris, France. Vermeulen, Rohrer, and Beverly were supported in part by the Laboratory for Telecommunication Sciences. Vermeulen, Fourmaux, and Friedman were supported in part by a university research grant from the French Ministry of Defense.

References

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

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

[3] 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.

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] Website for topology validation . http://heartbeat.planet-lab.eu:8000.

[15] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a large European IXP. In *Proc. ACM SIGCOMM '12*. ACM, 2012.

[16] Brice Augustin, Timur Friedman, and Renata Teixeira. Exhaustive path tracing with Paris traceroute. In *Proc. ACM CoNEXT '07*, 2006.

[17] Brice Augustin, Timur Friedman, and Renata Teixeira. Multipath tracing with Paris traceroute. In *Proc. E2EMON '07*, 2007.

[18] Brice Augustin, Timur Friedman, and Renata Teixeira. Measuring multipath routing in the Internet. *IEEE/ACM Transactions on Networking*, 19(3):830–840, June 2011.

[19] Robert Beverly. Yarrp'ing the Internet: Randomized high-speed active topology discovery. In *Proc. ACM IMC '16*, 2016.

[20] Robert Beverly, Ramakrishnan Durairajan, David Plonka, and Justin P Rohrer. In the IP of the beholder: Strategies for active IPv6 topology discovery. In *Proc. ACM IMC '18*, 2018.

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

[22] Kenneth L Calvert, Matthew B Doar, and Ellen W Zegura. Modeling Internet topology. *IEEE Communications magazine*, 35(6):160–163, 1997.

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

[24] kc claffy, Young Hyun, Ken Keys, Marina Fomenkov, and Dmitri Krioukov. Internet mapping: From art to science. In *IEEE DHS Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, 2009.

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

[26] Ítalo Cunha, Renata Teixeira, and Christophe Diot. Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing. In *Proc. PAM '11*, 2011.

[27] Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. DTRACK: A system to predict and track Internet path changes. *IEEE/ACM Transactions on Networking*, 22(4):1025–1038, August 2014.

[28] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM Computer Communication Review*, 29(4):251–262, 1999.

[29] Christian E Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992 (Informational), November 2000.

[30] Van 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.

[31] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP geolocation using delay and topology measurements. In *Proc. ACM IMC '06*, 2006.

[32] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize CDN performance. In *Proc. ACM IMC '09*, 2009.

[33] M. Luckie, B. Huffaker, k. claffy, A. Dhamdhere, and V. Giotsas. AS relationships, customer cones, and validation. In *Proc. ACM IMC '13*, 2013.

[34] Matthew Luckie. Scamper: A scalable and extensible packet prober for active measurement of the Internet. In *Proc. ACM IMC '10*, 2010.

[35] Matthew Luckie and Robert Beverly. The impact of router outages on the AS-level Internet. In *Proc. ACM SIGCOMM '17*, 2017.

[36] Matthew Luckie, Young Hyun, and Bradley Huffaker. Traceroute probe method and forward IP path inference. In *Proc. ACM IMC '08*, 2008.

[37] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, Jonathan M Smith, et al. Pushing the boundaries with bdrmapIT: Mapping router ownership at Internet scale. In *Proc. ACM IMC '18*, 2018.

[38] Vern Paxson. End-to-end routing behavior in the internet. *IEEE/ACM transactions on Networking*, 5(5):601–615, 1997.

[39] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999.

[40] Riccardo Ravaioli, Guillaume Urvoy-Keller, and Chadi Barakat. Characterizing ICMP rate limitation on routers. In *Proc. ICC '15*, 2015.

[41] RIPE NCC. RIPE Atlas, 2019. `https://atlas.ripe.net/`.

[42] 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`.

[43] James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, Qian Shi, and Justin P. Rohrer. Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation. *Springer Telecommunication Systems*, 52(2):705–736, February 2011.

[44] Darryl Veitch, Brice Augustin, Renata Teixeira, and Timur Friedman. Failure control in multipath route tracing. In *Proc. IEEE Infocom '09*, 2009.

[45] Kevin Vermeulen, Stephen D Strowes, Olivier Fourmaux, and Timur Friedman. Multilevel MDA-Lite Paris traceroute. In *Proc. ACM IMC '18*. ACM, 2018.

[46] Walter Willinger, David Alderson, and John 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.

[47] Mingyang Zhang, Radhika Niranjan Mysore, Sucha Supittayapornpong, and Ramesh Govindan. Understanding Lifecycle Management Complexity of Datacenter Topologies. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019.

[48] Zheng Zhang, Ying Zhang, Y. Charlie Hu, Z. Morley Mao, and Randy 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.