Reverse traceroute

Ethan Katz-Bassett[*] *Harsha V. Madhyastha*[†] *Vijay Kumar Adhikari*[‡] *Colin Scott*[*]
Justine Sherry[*] *Peter van Wesep*[*] *Thomas Anderson*[*] *Arvind Krishnamurthy*[*]

Abstract

Traceroute is the most widely used Internet diagnostic tool today. Network operators use it to help identify routing failures, poor performance, and router misconfigurations. Researchers use it to map the Internet, predict performance, geolocate routers, and classify the performance of ISPs. However, traceroute has a fundamental limitation that affects all these applications: it does not provide reverse path information. Although various public traceroute servers across the Internet provide some visibility, no general method exists for determining a reverse path from an arbitrary destination.

In this paper, we address this longstanding limitation by building a reverse traceroute system. Our system provides the same information as traceroute, but for the reverse path, and it works in the same case as traceroute, when the user may lack control of the destination. We use a variety of measurement techniques to incrementally piece together the path from the destination back to the source. We deploy our system on PlanetLab and compare reverse traceroute paths with traceroutes issued from the destinations. In the median case our tool finds 87% of the hops seen in a directly measured traceroute along the same path, versus only 38% if one simply assumes the path is symmetric, a common fallback given the lack of available tools. We then illustrate how we can use our reverse traceroute system to study previously unmeasurable aspects of the Internet: we present a case study of how a content provider could use our tool to troubleshoot poor path performance, we uncover more than a thousand peer-to-peer AS links invisible to current topology mapping efforts, and we measure the latency of individual backbone links with average error under a millisecond.

1 Introduction

Traceroute is a simple and widely used Internet diagnostic tool. It measures the sequence of routers from the source to the destination, supplemented by round-trip delays to each hop. Operators use it to investigate routing failures and performance problems [39]. Researchers use it as the basis for Internet maps [1, 22, 34], path prediction [22], geolocation [42, 14], ISP performance analysis [25], and anomaly detection [46, 19, 44, 43].

However, traceroute has a fundamental limitation – it provides no reverse path information, despite the fact that policy routing and traffic engineering mean that paths are generally asymmetric [15]. As Richard Steenbergen, CTO for nLayer Communications, put it at a recent tutorial for network operators on troubleshooting, "the number one go-to tool is traceroute," but "asymmetric paths [are] the number one plague of traceroute" because "the reverse path itself is completely invisible" [39].

This invisibility hinders operators. For instance, although Google has data centers distributed around the world, 20% of client prefixes experience unreasonably high latency, even with a nearby server. In working with a Google group trying to improve this performance, we found that we would have been able to more precisely troubleshoot problems if we could measure the path from clients back to Google [21].

Similarly, the lack of reverse path information restricts researchers. Traceroute's inability to measure reverse paths forces unrealistic assumptions of symmetry on systems with goals ranging from path prediction [22], geolocation [42, 14], ISP performance analysis [25], and prefix hijack detection [46]. Recent work shows that measured topologies miss many of the Internet's peer-to-peer links [29, 16] because mapping projects [1, 22, 34] lack the ability to measure paths from arbitrary destinations.

Faced with this shortcoming with the traceroute tool, operators and researchers turn to various limited workarounds. Surprisingly, network operators often resort to posting problems on operator mailing lists asking others to issue traceroutes to help diagnosis [30, 41]. Public web-accessible traceroute servers hosted at various locations around the world provide some help, but their numbers are limited. Without a server in every network, one cannot know whether any of those available have a path similar to the one of interest. Further, they are not intended for the heavy load incurred by regular monitoring. A few modern systems attempt to deploy traceroute clients on end-user systems around the world [34, 9], but none of them are close to allowing an arbitrary user to trigger an on-demand traceroute towards the user from anywhere in the world.

Our goal is to address this basic restriction of traceroute by building a tool to provide the same basic information as traceroute – IP-address-level hops along the path, plus round-trip delay to each – but along the reverse path from the destination back to the source. We have implemented our reverse traceroute system and make

[*]Dept. of Computer Science, Univ. of Washington, Seattle.
[†]Dept. of Computer Science, Univ. of California, San Diego.
[‡]Dept. of Computer Science, Univ. of Minnesota.

it available at `http://revtr.cs.washington.edu`. While traceroute runs as a stand-alone program issuing probes on its own behalf, ours is a distributed system comprised of a few tens to hundreds of vantage points, owing to the difficulty in measuring reverse paths. As with traceroute, our reverse traceroute tool does not require control of the destination, and hence can be used with arbitrary targets. All our tool requires of the target destination is an ability to respond to probes, the same requirement as standard traceroute. It does not require new functionality from routers or other network components.

Our system builds a reverse path incrementally, using a variety of methods to measure reverse hops, and stitching them together into a path. We combine the view of multiple vantage points to gather information unavailable from any single one. We start by measuring the paths from the vantage points to the source. This limited atlas of a few hundred or thousand routes to the source serves to bootstrap the rest of our measurements, allowing us to measure the path from an arbitrary destination by building back the path from the destination until it intersects the atlas. We use three main measurement techniques to build backwards. First, we rely on the fact that Internet routing is generally destination-based, allowing us to piece together the path one hop at a time. Second, we employ the IP timestamp and record route options to identify hops along the reverse path. Third, we use limited source spoofing – spoofing from one vantage point as another – to use the vantage point in the best position to make the measurement. This controlled spoofing allows us to overcome many of the limitations inherent in using IP options [36, 35, 13], while remaining safe, as the spoofed source address is one of our hosts. Just as many projects use traceroute, others have used record route and spoofing for other purposes. Researchers used record route to identify aliases and generate accurate topologies [35], and our earlier work used spoofing to characterize reachability problems [19]. In this work, we are the first to show that the combination of these techniques can be used to measure arbitrary reverse paths.

Experienced users realize that, while traceroute is useful, it has numerous limitations and caveats, and can be potentially misleading [39]. Similarly, our tool has limitations and caveats. Section 5.1 includes a thorough discussion of how the output of our tool might differ from a direct traceroute from the destination to the source, as well as how both might differ from the actual path traversed by traffic. Just as traceroute provides little visibility when routers do not send TTL-expired messages, our technique relies on routers honoring IP options. When our measurement techniques fail to discern a hop along the path, we fall back on assuming the hop is traversed symmetrically; our evaluation results show that, in the median (mean) case for paths between PlanetLab sites, we measure 95% (87%) of hops without assuming symmetry. The need to assume symmetry in cases of an unresponsive hop points to a limitation of our tool compared to traceroute; whereas traceroute can often measure past an unresponsive hop or towards an unreachable destination, our tool must sometimes guess that it is measuring the proper path.

We rely on routers to be "friendly" to our techniques, yet some of our techniques have the potential for abuse and can be tricky for novices to use without causing disturbances. As we ultimately want our tool widely used operationally, we have attempted to pursue our approach in a way that encourages continued router support. Our system performs measurements in a way that emphasizes network friendliness, controlling probe rate across all measurements. We presented the work early on at NANOG [28] and RIPE [32] conferences, and so far the response from operators has been positive towards supporting our methods (including source spoofing). We believe the goal of wide use is best served by a single, coordinated system that services requests from all users.

We evaluate the effectiveness of our system as deployed today, though it should improve as we add vantage points. We find that, in the median (mean) case for paths between PlanetLab sites, our technique reveals 87% (83%) of the routers and 100% (94%) of the points-of-presence (PoPs), compared to a traceroute issued from the destination. Paths between public traceroute servers and PlanetLab show similar results. Because our technique requires software at the source, our evaluation is limited to paths back to PlanetLab nodes we control. We believe our reverse traceroute system can be useful in a range of contexts, and we provide three illustrative examples. We present a case study of how a content provider could use our tool to troubleshoot poor reverse path performance. We also uncover thousands of links at core Internet exchange points that are invisible to current topology mapping efforts. We use our reverse traceroute tool to measure link latencies in the Sprint backbone network with less than a millisecond of error, on average.

2 Background

In this section, we provide the reader some background on Internet routing and traceroute.

Internet routing: First, a router generally determines the route on which to forward traffic based only on the destination. With a few caveats such as load-balancing and tunneling, the route from a given point towards a particular destination is consistent for all traffic, regardless of its source. While certain tunnels may violate this assumption, best practices encourage tunnels that appear as atomic links. Second, asymmetry between forward and reverse paths stems from multiple causes. An AS is free to choose its next hop among the alternatives, whether or

not that leads to a symmetric route. Two adjacent ASes may use different peering points in the two directions due to policies such as early-exit/hot-potato routing. Even within an individual AS, traffic engineering objectives may lead to different paths.

Standard traceroute tool: Traceroute measures the sequence of routers from a source to a destination, without requiring control of the destination. When traceroute was originally developed, most paths were symmetric, but that assumption no longer holds. Traceroute works by sending a series of packets to the destination, each time incrementing the time-to-live (TTL) from an initial value of one, in order to get ICMP TTL exceeded responses from each router on the path in turn. Each response will have an IP address of an interface of the corresponding router. Additionally, traceroute measures the time from the sending of each packet to the receipt of the response, yielding a round-trip latency to each intermediate router. Because the destination resets the TTL in its reply, traceroute only works in the forward direction.

The path returned by traceroute is a feasible, but possibly inaccurate route. First, each hop comes from a response to a different probe packet, and the different probes may take different paths for reasons including contemporaneous routing changes or load balancing. The Paris traceroute customizes probe packets to provide consistent results across flow-based load balancers, as well as to systematically explore the load-balancing options [2]. For all our traceroutes, we use the Paris option that measures a single consistent path. Second, some routers on the path may not respond. For example, some routers may be configured to rate-limit responses or to not respond at all. Third, probe traffic may be treated differently than data traffic.

Despite these caveats, traceroute has proved to be extremely useful. Essential to traceroute's utility is its universality, in that it does not require anything of the destination other than an ability to respond to probe packets.

3 Reverse Traceroute

We seek to build a reverse path tool equivalent to traceroute. Like traceroute, ours should work universally without requiring control of a destination, and it should use only features available in the Internet as it exists today. The reverse traceroute tool should return IP addresses of routers along the reverse path from a destination back to the source, as well as the round-trip delay from those routers to the source.

At a high level, the source requests a path from our system, which coordinates probes from the source and from a set of distributed vantage points to discover the path. First, distributed vantage points issue traceroutes to the source, yielding an atlas of paths towards it (Fig. 1(a)). This atlas provides a rich, but limited in scope, view of how parts of the Internet route towards the source. We use this limited view to bootstrap measurement of the desired path. Because Internet routing is generally destination-based, we assume that the path to the source from any hop in the atlas is fixed (over short time periods) regardless of how any particular packet reaches that hop; once the path from the destination to the source reaches a hop in the atlas, we use the atlas to derive the remainder of the path. Second, using techniques we explain in Sections 3.1 and 3.2, we measure the path back from the destination incrementally until it intersects this atlas (Fig. 1(b)). Finally, as shown in an example in Section 3.3, we merge the two components of the path, the destination-specific part measured from the destination until it intersects the atlas, and the atlas-derived path from this intersection back to the source, to yield a complete path (Fig. 1(c)).

3.1 Identify Reverse Hops with IP Options

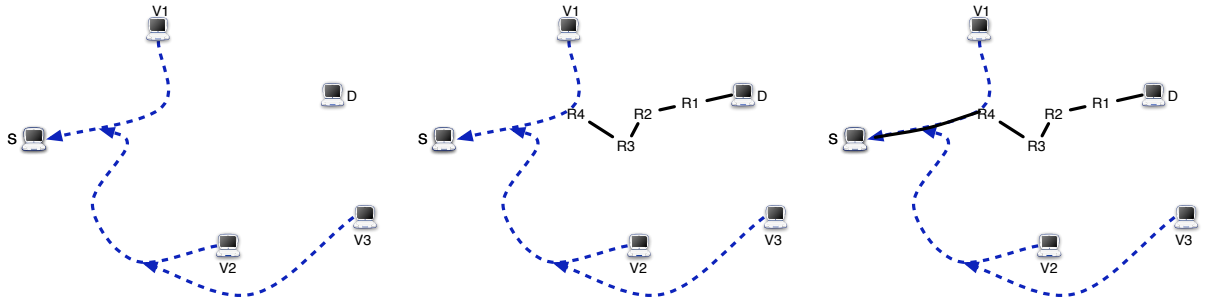
We use two basic measurement primitives, the Record Route and Timestamp IP options. While TTL values are reset by the destination, restricting traceroute to measuring only on the forward path, IP options are generally reflected in the reply from a destination, so routers along both the forward and reverse path process them. We briefly explain how the options work:

IP Record-route option (*RR*): With this option set, a probe records the router interfaces it encounters. The IP standard limits the number of recorded interfaces to 9; once those fill, no more interfaces are recorded.

IP timestamp option (*TS*): IP allows probes to query a set of specific routers for timestamps. Each probe can specify up to four IP addresses, in order; if the probe traverses the router matching the next IP address that has yet to be stamped, the router records a timestamp. The addresses are ordered, so a router will not timestamp if its IP address is in the list but is not the next one.

We use these options to gather reverse hops as follows:

- *RR-Ping*$(S \rightarrow D)$: As shown in Figure 2(a)), the source S issues an ICMP Echo Request (henceforth *ping*) probe to D with the RR option enabled. If RR slots remain when the destination sends its response, then routers on the reverse path will record some of that route. This allows a limited measurement of the reverse path, as long as the destination is fewer than 9 hops from the source.
- *TS-Query-Ping*$(S \rightarrow D|D, R)$: As shown in Figure 2(b)), the source S issues an ICMP ping probe to D with the timestamp query enabled for the ordered pair of IP addresses D and R. R will record its timestamp only if it is encountered by the probe after D has stamped the packet. In other words, if S receives a timestamp for R, then it knows R appears

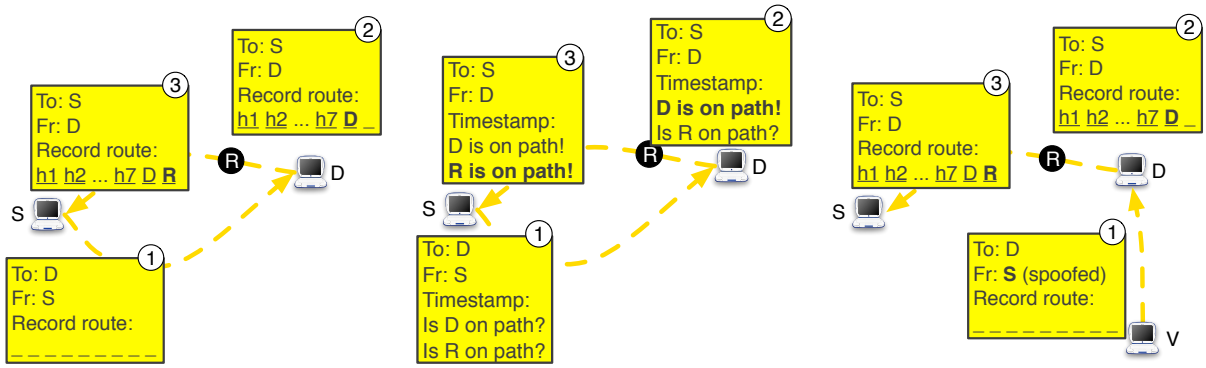


(a) Vantage points traceroute to S, creating an atlas of known paths.

(b) Vantage points measure path from D until it intersects a path in the atlas.

(c) Combine to yield complete path.

Figure 1: High-level overview of the reverse traceroute technique. We explain how to measure from D back to the atlas in § 3.1- 3.2.



(a) S sends a record-route ping. The header includes slots for 9 IP addresses to be recorded (1). If the packet reaches D with slots remaining, D adds itself (2), and routers on the reverse path fill the remaining slots (3).

(b) S sends a timestamp ping, asking first for D to provide a stamp if encountered, then for R to provide one (1). If D supports the timestamp option, it fills out a timestamp (2). Because the timestamp requests are ordered, R only fills out a timestamp if encountered after D, necessarily on the reverse path (3).

(c) Vantage point V sends a record-route ping to D, spoofing as S (1). D replies to S (2), allowing S to discover that R is on the reverse path (3). We use this technique when S is out of record-route range of D, but V is close enough.

Figure 2: Three measurement techniques that allow us to establish that R is on the reverse path from D back to S. In §4, we give two techniques, akin to (c), that use spoofing to overcome limitations in timestamp support.

on the reverse path. For our purposes, the value of the timestamp is meaningless; we just care whether or not a particular router processes the packet. Thus, if we guess a router on the return path, the TS option can confirm our hypothesis.

We use existing network topology information – specifically IP-level connectivity of routers from Internet mapping efforts [22] – to determine candidate sets of routers for the reverse path. Routers adjacent to D in the topology are potential next hops; we use timestamp query probes to check whether any of these potential next hops is on the path from D to S.

Note that there are some caveats to using the probes outlined above. One is that from each vantage point, only a fraction of routers will be reachable within record route's limit of 9 hops. Another is that some ISPs filter and drop probes with IP options set. Further, some routers do not process the IP options in the prescribed manner. Fortunately, we can overcome these limitations in the common case by carefully orchestrating the measurements from a diverse set of vantage points.

3.2 Spoof to Best Use Record Route

A source-spoofed probe (henceforth referred to as a spoofed probe) is one in which the prober sets the source address in the packet to one other than its own. We use a limited form of spoofing, where we replace the source address in a probe with the "true" source of the reverse traceroute. This form of spoofing is an extremely powerful measurement tool. When V probes D spoofing as S, D's response will go to S; we refer to V as the spoofer and S as the receiver. This method allows the probe to traverse the path from D to S without having to traverse the path from S to D and without having a vantage point in D's prefix. We could hypothetically achieve a similar probe trajectory using loose source routing (from V

to S, but source routed through D) [31]. However, a source-routed packet can be identified and filtered anywhere along the path, and such packets are widely filtered [3], too often to be useful in our application. On the other hand, if a spoofed packet is not ingress filtered near the spoofer, it thereafter appears as a normal packet; we can use a source capable of spoofing to probe along any path. Based on our measurements to all routable prefixes, many routers that filter packets with the source route option do not filter packets with the timestamp or record route options. This difference is likely because source routing can be used to violate routing policy, whereas the timestamp and record route options cannot.

This arrangement allows us to use the most advantageous vantage point with respect to the particular measurement we want to perform. Our earlier system Hubble used limited spoofing to check one-way reachability [19]; we use it here to overcome limitations of IP options. Without spoofing, RR's 9 IP address limit restricts it to being useful only when S is near the target. However, as shown in Figure 2(c),if some vantage point V is close enough to reach the target within 8 RR slots, then we can probe from V spoofing as S to receive IP addresses on the path back to S. Similarly, spoofing can bypass problematic ASes and machines, such as those that filter timestamp-enabled packets or those that do not correctly implement the option.

Although spoofing is often associated with malicious intent, we use it in a very controlled, safe fashion. A node requests a reverse path measurement, then receives responses to probes sent by vantage points spoofing as it. No harm can come from causing one of our own nodes to receive measurement packets. This form of spoofing shares a purpose with the address rewriting done by middleboxes such as NATs, controlling the flow of traffic to a cooperative machine, rather than with malicious spoofing which seeks concealment. Since some ISPs filter spoofed packets, we test from each host and only send further spoofed probes where allowed. We have been issuing spoofed probes for over two years without complaint.

Roughly 20% of PlanetLab sites allow spoofing; this ability is not limited to PlanetLab: the Spoofer project tested 12,000 clients and found that 31% could send spoof packets [5]. Even if filtering increases, we believe, based on positive feedback from operators, that the value of our service will encourage an allowance (supported by router ACLs) for a small number of measurement nodes to issue spoofed probes using a restricted set of ports. An even simpler approach is to have routers rate limit these spoofed options packets (just as with UDP probes) and filter spoofed probes sent to broadcast addresses, thereby reducing the security concerns without diminishing their utility for network measurements.

3.3 Incrementally Build Paths

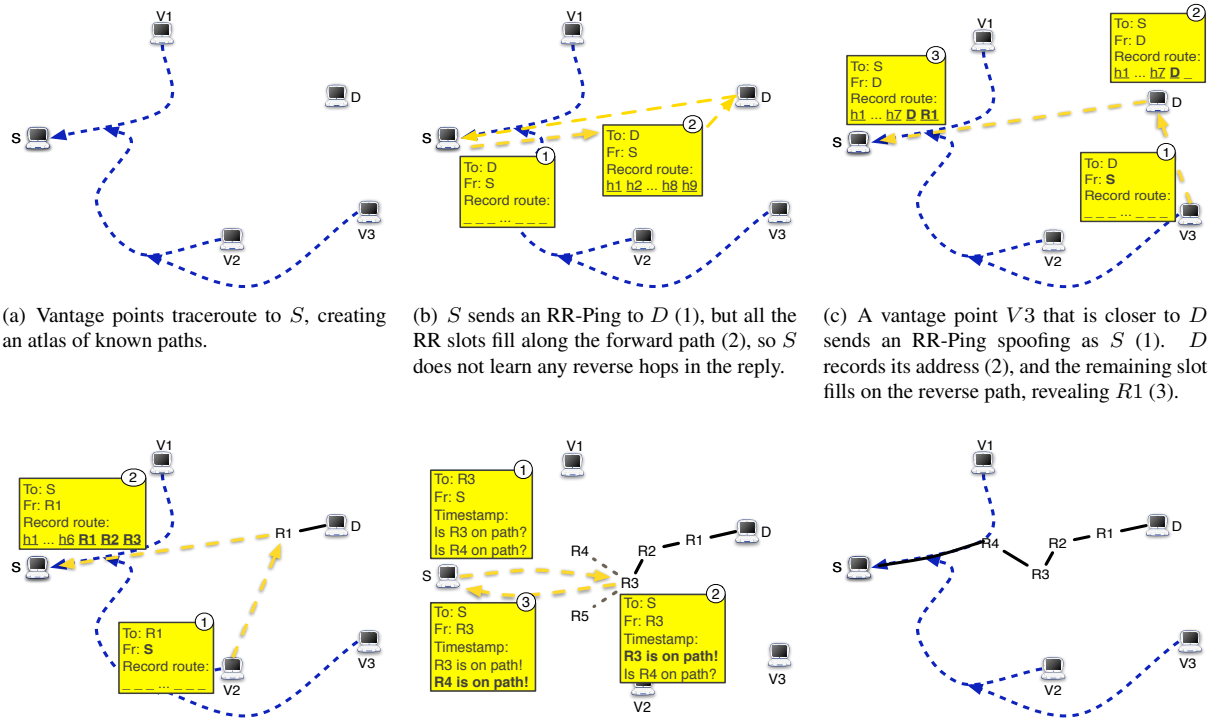
IP option-enabled probes, coupled with spoofing as S from another vantage point, give us the ability to measure a reverse hop from D on the path back to S. We can use the same techniques to stitch together a path incrementally – once we know the path from D goes through R, we need only determine the route at R towards S when attempting to discover the next hop. Because Internet routing is generally based on the destination, each intermediate router R we find on the path can become the new destination for a reverse traceroute back to the source. Further, if R is on a path from some vantage point V to S, then we can infer the rest of D's return path from R onward as being the same as V's. This assumption holds even in cases of packet-, flow-, and destination-based load balancing, so long as R balances traffic independently of other routers and of the source.

Figure 3 illustrates how we can compose the above set of techniques to determine the reverse path from D to S, when we have control over S and a set of other vantage points (V_1, V_2, V_3). We assume that we have a partial map of router-level connectivity, e.g., from a traditional offline mapping effort.

We begin by having the vantage points issue traceroute probes to S (Figures 1(a) and 3(a)). These serve as a baseline set of observed routes towards S that can be used to complete a partially inferred reverse path. We then issue *RR-Ping*$(S \rightarrow D)$ to determine if the source S is within 8 RR hops of the destination, i.e., whether a ping probe from S can reach D without filling up its entire quota of 9 RR hops (Figure 3(b))[1]. If the source is within 8 RR hops of D, this probe would determine at least the first hop on the reverse path, with further hops recovered in an iterative manner.

If the source is not within 8 hops, we determine whether some vantage point is within 8 RR hops of D (Section 4.5 describes how we do this). Let V_3 be one such vantage point. We then issue a spoofed RR ping probe from V_3 to D with the source address set to S (Figure 3(c)). This probe traverses the path $V_3 \rightarrow D \rightarrow S$ and records IP addresses encountered. The probe reveals R_1 to be on the reverse path from D to S. We then iterate over this process, with the newly found reverse hop as the target of our probes. For instance, we next determine a vantage point that is within 8 RR hops of R_1, which could be a different vantage point V_2. We use this new vantage point to issue a spoofed RR ping probe to determine the next hop on the reverse path (Figure 3(d)).

[1]This is not quite as simple as sending a TTL=8 limited probe, because of issues with record route implementations [35]. Some routers on the forward path might not record their addresses, thereby freeing up more slots for the reverse path, while some other routers might record multiple addresses or might record their address but not decrement or respond to TTL-limited probes.



(a) Vantage points traceroute to S, creating an atlas of known paths.

(b) S sends an RR-Ping to D (1), but all the RR slots fill along the forward path (2), so S does not learn any reverse hops in the reply.

(c) A vantage point $V3$ that is closer to D sends an RR-Ping spoofing as S (1). D records its address (2), and the remaining slot fills on the reverse path, revealing $R1$ (3).

(d) A vantage point $V2$ close to $R1$ sends an RR-Ping spoofing as S (1), discovering $R2$ and $R3$ on the reverse path (2).

(e) We use an Internet map to find routers adjacent to $R3$, then send each a TS-Query-Ping to verify which is on the reverse path.

(f) When we intersect a known path in our traceroute atlas, we assume the rest of the path from D follows that route.

Figure 3: Illustration of the incremental construction of a reverse path using diverse information sources.

In some cases, a single RR ping probe may determine multiple hops, as in the illustration with R_2 and R_3.

Now, consider the case where neither S nor any of the vantage points is within 8 hops of R_3. In that case, we consider the potential next hops to be routers adjacent to R_3 in the known topology. We issue timestamp probes to verify whether the next hop candidates R_4 and R_5 respond to timestamp queries *TS-Query-Ping*$(S \rightarrow D|D, R_4)$ and *TS-Query-Ping*$(S \rightarrow D|D, R_5)$ (as shown in Figure 3(e)). When R_4 responds, we know that it is adjacent to R_3 in the network topology and is on the reverse path from R_3, and so we assume it is the next hop on the reverse path. We continue to perform incremental reverse hop discovery until we intersect with a known path from a vantage point to S [2], at which point we consider that to be the rest of the reverse path (Figures 1(c) and 3(f)). Once the procedure has determined the hops in the reverse path, we issue pings from the source to each hop in order to determine the round-trip latencies.

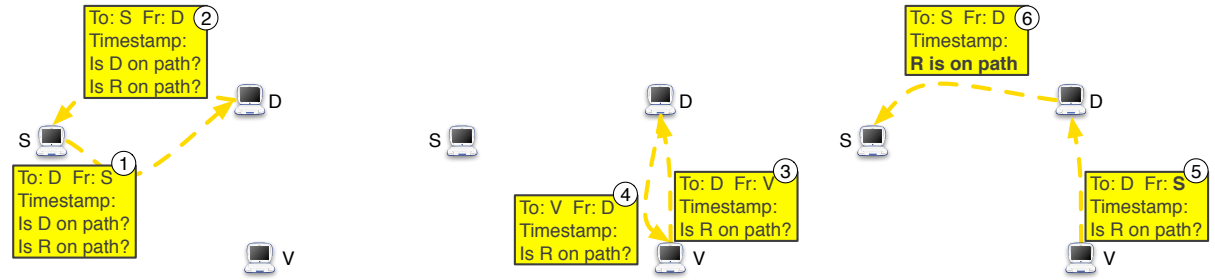
Sometimes, we may be unable to measure a reverse hop using any of our techniques, but we still want to provide the user with useful information. When reverse traceroute is unable to calculate the next hop in a path, the source issues a standard traceroute to the last known hop on the path. We then assume the last link is traversed symmetrically, and we try to calculate the rest of the reverse path from there. In Section 5.1 and Section 5.2, we present results that show that we usually do not have to assume many symmetric hops and that, even with this approximation, we still achieve highly accurate paths.

[2]Measurement techniques may discover different addresses on a router [36], so we determine intersections using alias data from topology mapping projects [20, 22, 35] and a state-of-the-art technique [4].

4 System Implementation

Section 3 describes how our techniques in theory would allow us to measure a reverse path. In this section, we discuss how we had to vary from that ideal description in response to realities of available vantage points and of router implementations. In addition, the following goals drive our system design:

- *Accuracy:* It should be robust to variations in how options-enabled packets are handled by routers.
- *Coverage:* The system should work for arbitrary destinations irrespective of ISP-specific configurations.
- *Scalability:* It needs to be selective with the use of vantage points and introduce as little measurement traffic as possible.



(a) S sends a timestamp ping to D (1) and receives a reply, but D has not filled out a timestamp (2).

(b) We find a V s.t., when V pings D asking for R's timestamp (3), it does not receive a stamp (4). This response indicates that R is not on V's path to D.

(c) V spoofs as S, pinging D (5), and S receives a timestamp for R (6). Because we established that R is not on V's path to D, R must be on the reverse path from D to S.

Figure 4: Example of how we discover a reverse hop with timestamp even though D does not stamp, as long as it replies.

4.1 Architecture

Our system consists of vantage points (VPs), which issue measurements, a controller, which coordinates the VPs to measure reverse paths, and sources, which request paths to them. We use a local machine at UW as a controller. When a VP starts up, it registers with the controller, which can then send it probe requests. A source runs our software to issue standard traceroutes, *RR-Pings*, and *TS-Query-Pings*, and to receive responses to probes from VPs spoofing as the source. However, it need not spoof packets itself. Currently, our source software only runs on Linux and (like the ICMP traceroute option `traceroute -I`) requires root permission. The controller receives requests from sources and combines the measurements from VPs to report reverse path information. While measuring a reverse traceroute, the controller queues up all incoming requests. When the ongoing measurement completes, the controller serves all requests in the queue as a batch, in synchronized rounds of probes. This design lets us carefully control the rate at which we probe any particular router, as well as to reuse measurements when a particular source requests multiple destinations.

We use topology maps from iPlane [22][3] to identify adjacent routers to test with *TS-Query-Pings* (Fig. 3(e)). To increase the set of possible next-hop routers, we consider the topology to be the union of maps from the previous 2 weeks. Since we verify the reverse hops using option-enabled pings, stale topology information makes our system less efficient but does not introduce error.

4.2 Current Deployment

Our current deployment uses one host at each of the more than 200 active PlanetLab sites as VPs to build an atlas of traceroutes to a source (Fig. 3(a)). Over the course of our study, 60+ PlanetLab sites allowed spoofed probes at least some of the time. We employ one host at each of these sites as spoofing nodes. Routers upstream from the other PlanetLab sites filter spoofed probes, so we did not spoof from them. We also use one host at each of 14 Measurement Lab sites [26], most of which allow spoofing. Various organizations provide public web-accessible traceroute servers, and we employ 1200 of them [3]. These nodes issue only traceroutes and cannot set IP options, spoof, or receive spoofed probes. We use them to expand the sets of known paths to our sources.

We have currently tested our client software only on PlanetLab (Linux) machines. We make it available as a demo; our website `http://revtr.cs.washington.edu` allows users to enter an IP address and measure the reverse path back from it to a PlanetLab node. Packaging the code for widespread deployment is future work. Because we have dozens of operators asking to use the system, we are being patient to avoid a launch that does not perform up to expectations.

4.3 Correcting for Variations in IP Options Support

We next explain how we compensate for variations in support for the timestamp option. When checking if R is on the reverse path from D, we normally ask for both D's and R's timestamp, to force R to only stamp on the reverse path. However, we found that, of the addresses in a day's iPlane atlas that respond to ping, 16.6% of the addresses respond to timestamp-enabled pings, but do not stamp, so we cannot use that technique to know that R stamped on the reverse path. Figure 4 illustrates how we use spoofing to address this behavior. Essentially, we find a VP V which we can establish does not have R on its path to D, then V pings D spoofing as S, asking for R's timestamp (but not D's). If S receives a stamp for R, it proves R is on the reverse path from D. This technique will not work if all vantage points have R on their paths. We examined iPlane traceroutes to destinations in 140,000 prefixes and found at least two adjacent hops for 55% of destinations.

[3] iPlane issues forward traceroutes from PlanetLab sites and traceroute servers to around 140K prefixes.

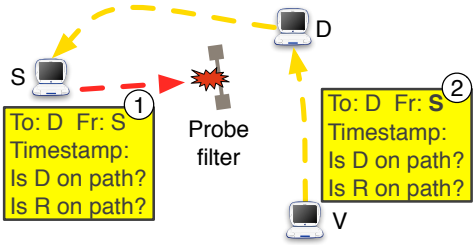


Figure 5: If a timestamp probe from S encounters a filter (1), we can often bypass it by spoofing as S from a different vantage point (2), as long as the filter is just on the forward path.

4.4 Avoiding Probe Filters to Improve Coverage

We next discuss techniques to improve the coverage of our measurements. Some networks may filter ICMP packets, and others filter packets with options enabled. In the course of measuring a reverse path, if a source attempts a TS or RR measurement and does not receive a response, we retry the measurement with a VP spoofing as the source. As seen in Figure 5, if filtering occurs only along the source's forward path, and the VP's forward path does not have a filter, the original source should receive the response.

We demonstrate the effectiveness of this approach on a small sample of 1000 IP addresses selected at random out of those in the iPlane topology known to respond to timestamp probes. The 1000 destinations include addresses in 662 ASes. We chose 10 spoofing PlanetLab vantage points we found to receive (non-spoofed) timestamp responses from the highest number of destinations, plus one host at each of the 209 working non-spoofing PlanetLab site. First, each non-spoofing node sent a series of timestamp pings to each destination; redundant probes account for loss due to something other than permanent filters. Of the 209 hosts, 103 received responses from at least 700 destinations; we dropped them from the experiment, as they do not experience significant filtering. Then, each spoofing vantage point sent 106 timestamp pings to each destination, spoofing as each of the remaining PlanetLab hosts in turn. Of these, 63 failed to receive any responses to either spoofed or non-spoofed probes; they are completely stuck behind filters or were not working. For the remaining 43 hosts, Figure 6 shows how many destinations each host receives responses from, both without and with spoofing. Our results show that some sites benefit significantly. In reverse traceroute's timestamp measurements, whenever the source does not receive a response, we retry with 5 spoofers. Since some vantage points have filter-free paths to most destinations, we use the 5 best overall, rather than choosing per destination. For the nodes that experience widespread filtering, spoofing enables a significant portion to still use timestamps as part of reverse traceroute. As we show in Section 5.2, our timestamp techniques help the overall coverage of the tool.

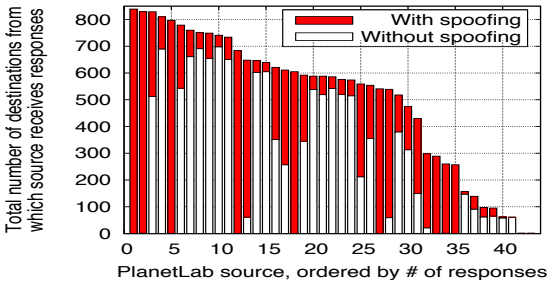


Figure 6: For 43 PlanetLab nodes, the number of destinations (out of 1000) from which the node receives timestamp responses. The graph shows the total number of unique destinations when sending the ping directly and then when also using 10 spoofers. The nodes are ordered by the total number of responding destinations. Other PlanetLab sites were tested but are not included in the graph: 103 did not experience significant filtering and 63 did not receive responses even with spoofing.

4.5 Selective Use of Vantage Points for Scalability

With spoofed RR, only nearby spoofers can find reverse hops, since each packet includes only 9 slots. Because many routers rate limit after only a few probes, we cannot send from many vantage points at once, in the hopes that one will prove close enough – the router might drop the probes from the VPs within range. Our goal is to determine which VPs are likely to be near a router before we probe it. Because Internet routing is generally based on the destination's prefix, a VP close to one address in a prefix is likely close to other addresses in the prefix.

Each day, we harvest the set of router IP addresses seen in the Internet atlas gathered by iPlane on the previous day and supplement the set with a recent list of pingable addresses [17]. Each day, every VP issues a record route ping to every address in the set. For each address, we determine the set of VPs that were near enough to discover reverse hops. We use this information in two ways during a reverse traceroute. First, if we encounter one of the probed addresses, we know the nearest VP to use. Second, if we encounter a new address, the offline probes provide a hint: the group of VPs within range of some address in the same prefix. Selecting the minimal number of vantage points to use from this group is an instance of the well known set cover optimization problem. We use the standard greedy algorithm to decide which VPs to use for a prefix, ordered by the number of additional addresses they cover within the prefix.

For a representative day, Figure 7 shows the coverage we achieve at given numbers of VPs per prefix. Our system determines the covering VPs for all prefixes, but the graph only includes prefixes for which we probed at least 15 addresses, as it is trivial to cover small prefixes. We see that, for most prefixes, we only need a small number of VPs. For example, in the median case, a single VP suffices for over 95% of addresses in the prefix, and we rarely need more than 4 VPs to cover the entire prefix.

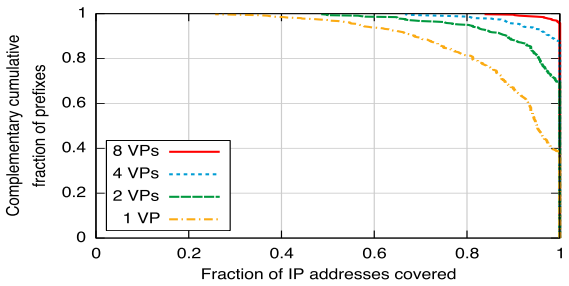


Figure 7: For prefixes in which iPlane observed $\geq$ 15 addresses, the fraction of the addresses for which we can find reverse hops using RR probes from a given number of vantage points per prefix. Note that we only include addresses within range of at least one vantage point. Prefixes with few addresses are trivial to cover using a small number of vantage points, so the graph excludes them to clearly show that we still only need a small number for most prefixes.

5 Evaluation

To test how well our reverse traceroute system can determine reverse paths, we consider evaluation settings that allow us to compare a reverse traceroute from D to S to a direct traceroute from D to S. A complete evaluation of the accuracy of our technique would require ground truth information about the path back from the destination. Obviously, we lack ground truth for the Internet, but we use two datasets, one PlanetLab-based and one using public traceroute servers, in which we can compare to a traceroute from D. For the reverse traceroute, we assume we do not control D and must measure the path using the techniques described in this paper. For the direct traceroute, we do control D and can simply issue a standard traceroute from D to S.

In the PlanetLab set, we employ as sources a host at each of 11 PlanetLab sites chosen at random from the spoofing nodes. As destinations, we use one host at each of the 200 non-spoofing PlanetLab sites that were working. Although such a set is not representative of the entire Internet, the destinations includes hosts in 35 countries. The measured reverse paths traversed 13 of the 14 transit-free commercial ISPs. Previous work observed route load balancing in many such networks [2], providing a good test for our techniques.

In the traceroute server set, we employ as sources a host at 10 of the same PlanetLab sites (one had gone down in the meantime). The 1200 traceroute servers we utilize belong to 186 different networks (many of which offer multiple traceroute servers with different locations). For each source, we choose a traceroute server at random from each of the 186 networks. We then issue a traceroute from the server to the PlanetLab source. Because in many cases we do not know the IP address of the traceroute server, we use the first hop along its path as the destination in our reverse traceroute measurements. When measuring a reverse traceroute from this destination back to the source, we exclude from our system all traceroute servers in the same network, to avoid providing our system with such similar paths as to make its task trivial.

5.1 Accuracy

How similar are the hops on a reverse traceroute to a direct traceroute from the destination back to the source? For the PlanetLab dataset, the *RevTR* line in Figure 8 depicts the fraction of hops seen on the direct traceroute that are also seen by reverse traceroute. Figure 9 shows the same for the traceroute server dataset. Note that, outside of this experimental setting, we would not normally have access to the direct traceroute from the destination. Using alias data from topology mapping projects [20, 22, 35] and aliases we discover using a state-of-the-art technique [4], we consider a traceroute hop and a reverse traceroute hop to be the same if they are aliases for the same router. We need to use alias information because the techniques may find different IP addresses on the same router [36]. For example, traceroute generally finds the ingress interface, whereas record route often returns the egress or loopback address. However, alias resolution is an active and challenging research area, and faulty aliases in the data we employ could lead us to falsely label two hops as equivalent. Conversely, missing aliases could cause us to label as different two interfaces on the same router. Because the alias sets we use are based on measurements from PlanetLab or similar vantage points, we likely have more complete alias data for our PlanetLab dataset than for our traceroute server dataset.

Using the available alias data, we find that the paths measured by our technique are quite similar to those seen by traceroute. In the median (mean) case, we measure 87% (83%) of the hops in the traceroute for the PlanetLab dataset. For the traceroute server dataset, we measure 75% (74%) of the hops in the direct traceroute, but 28% (29%) of the the hops discovered by reverse traceroute do not appear in the corresponding traceroute.

The figures also compare reverse traceroute to other potential ways of estimating the reverse path. All techniques used the same alias resolution. Researchers often (sometimes implicitly) assume symmetry, and operators likewise rely on forward path measurements when they need reverse ones. The *Guess Fwd* lines depict how many of the hops seen in a traceroute from R to S are also seen in a traceroute from S to R. In the median (mean) case, the forward path shares 38% (39%) of the reverse path's hops for the PlanetLab dataset and 40% (43%) for the traceroute server dataset. Another approach would be to measure traceroutes from a set of vantage points to the source. Using iPlane's PlanetLab

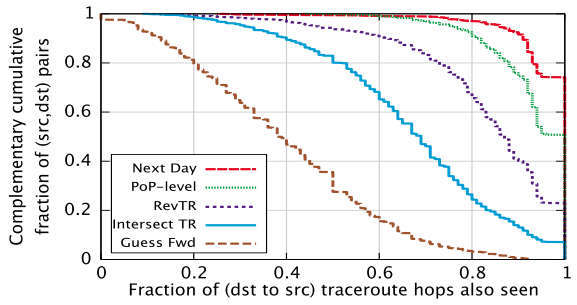


Figure 8: For reverse traceroute and techniques for approximating the reverse path, the fraction of hops on a direct traceroute from the destination to the source that the technique also discovers. Uses our PlanetLab dataset (reverse paths from 200 PlanetLab destinations back to 11 PlanetLab sources). [Key labels are in the same top-to-bottom order as the lines.]

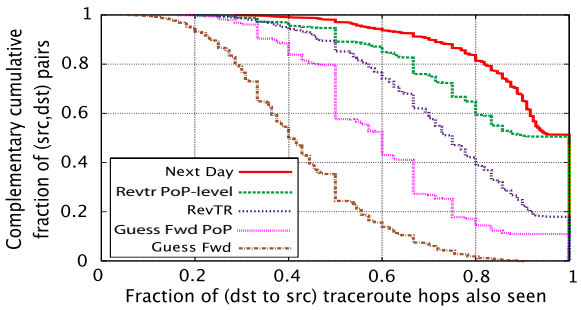


Figure 9: For reverse traceroute and techniques for approximating the reverse path, the fraction of hops on a direct traceroute from the destination to the source that the technique also discovers. Our traceroute server dataset includes reverse paths from servers in 186 networks back to 10 PlanetLab sources. [Key labels are in the same top-to-bottom order as the lines.]

and traceroute server measurements, the *Intersect TR* line in Figure 8 shows how well this approach works, by assuming the reverse path is the same as the forward path until it intersects one of the traceroutes [4]. No system today performs this type of path intersection on-demand for users. In the median (mean) case, this traceroute intersection shares 69% (67%) of the actual traceroute's hops. This result suggests that simply having a few hundred or thousand traceroute vantage points is not enough to reliably infer reverse paths; our system uses our novel measurement techniques to build off these traceroutes and achieve much better results.

What are the causes of differences between a reverse traceroute and a directly measured traceroute? Although it is common to think of the path given by traceroute as the true path, in reality it is also subject to measurement error. In this section, we discuss reasons traceroute and reverse traceroute may differ from each other and/or from the true path taken.

Assumptions of symmetry: When reverse traceroute is unable to identify the next reverse hop, we resort to assuming that hop is symmetric. These assumptions may lead to inaccuracies. For the PlanetLab dataset, if we consider only cases when we measure a complete path without assuming symmetry, in the median (mean) case reverse traceroute matches 93% (90%) of the traceroute alias-level hops. Similarly, for the traceroute server dataset, in the median (mean) case reverse traceroute finds 83% (81%) of the traceroute hops. We discuss how often we have to assume symmetry in Section 5.2.

Incomplete alias information: Many of the differences between the paths found by reverse traceroute and traceroute are due to missing alias information. Most alias-pair identification relies on sending probes to the two IP addresses and comparing the IP-IDs of the responses. For the PlanetLab dataset, of all the *missing* addresses seen in a traceroute that are not aliases of any hop in the corresponding reverse traceroute, 88% do not allow for such alias resolution [4]. Similarly, of all *extra* addresses seen in some reverse traceroute that are not aliases of any hop in the corresponding reverse traceroute, 82% do not allow for alias resolution. For the traceroute server dataset, 75% of the missing addresses and 74% of the extra ones do not allow it. Even for addresses that do respond to alias techniques, our alias sets are likely incomplete.

In these cases, it is possible or even likely that the two measurement techniques observe IP addresses that appear different but are in fact aliases of the same router. To partially examine how this lack of alias information limits our comparison, we use iPlane's Point-of-Presence (PoP) clustering, which maps IP addresses to PoPs defined by (AS,city) pairs [22]. For many applications such as diagnosis of inflated latencies [21], PoP level granularity suffices. iPlane has PoP mappings for 71% of the missing addresses in the PlanetLab dataset and 77% of the extra ones. For the traceroute server dataset, for which we have less alias information, it has mappings for 79% of the missing addresses and 86% of the extra ones. Figures 8 and 9 include *PoP-Level* lines showing the fraction of traceroute hops seen by reverse traceroute, if we consider PoP rather than router-alias-level comparison. In the median case, the reverse traceroute includes all the traceroute PoPs in both graphs (mean=94%, 84%). If reverse traceroute were measuring a different path than traceroute, then one would expect PoP-level comparisons to differ about as much as alias-level ones. The implication of the measured PoP-level similarity is that, when traceroute and reverse traceroute differ, they usually differ only in which router or interface in a PoP the path traverses. As a point of comparison, Figure 9 includes a PoP-level version of the *Guess Fwd* line; in the median (mean) case, it includes only 60% (61%) of the PoPs; the paths are quite asymmetric even at the PoP granularity.

[4] We omit the line from Figure 9 to avoid clutter.

Load-balancing and contemporaneous path changes: Another measurement artifact is that traceroute and reverse traceroute may uncover different, but equally valid, paths, either due to following different load-balanced options or due to route changes during measurement. To partly capture these effects, the *Next Day* lines in Figure 8 and 9 compare how many of the traceroutes' hops are also on traceroutes issued the following day. For the PlanetLab dataset, 26% of the paths exhibit some router-level variation from day to day. For the traceroute dataset, 49% of paths changed at the router level and (not shown in the graph) 15% changed at the PoP-level. In a loose sense, these results suggest an upper bound – even the same measurement issued at a different time may yield a different path.

Hidden or anonymous routers: Previous work comparing traceroute to record route paths found that 16% of IP addresses appear with RR but do not appear in traceroutes [36, 35]. *Hidden* routers, such as those inside some MPLS tunnels, do not decrement TTL. *Anonymous* routers decrement TTL but do not send ICMP replies, appearing as '*' in traceroute.

Exceptional handling of options packets: Packets with IP options are generally diverted to a router's route processor and may be processed differently than on the line card. For example, previous work suggests that packets with options are load-balanced per-packet, rather than per-flow [35].

An additional source of discrepancies between the two techniques is that traceroute and reverse traceroute make different assumptions about routing. Our techniques assume destination-based routing – if the path from D to S passes through R, from that point on it is the same as R's path to S. An options packet reports only links it actually traversed. With traceroute, on the other hand, a different packet uncovers each hop, and it assumes that if $R1$ is at hop k and $R2$ is at hop $k+1$, then there is a link $R1$–$R2$. However, it does not make the same assumption about destination routing, as each probe uses (S,D) as the source and destination. These differing assumptions lead to two more causes of discrepancies between a traceroute and a reverse traceroute:

Traceroute inferring false links: Although we use the Paris traceroute technique for accurate traversal of flow-based load balancers, it can still infer false links in the case of packet-based balancing [2]. These spurious links appear as discrepancies between traceroute and reverse traceroute, but in reality show a limitation of traceroute.

Exceptions to destination-based routing: With many tunnels, an option-enabled probe will see the entire tunnel as a single hop. With certain tunnels, however, our assumption of destination-based routing may not hold. When probed directly, an intermediate router inside the tunnel may use a path to the destination other than the one that continues through the tunnel. To partly capture the degree of this effect, we perform a study that eliminates it. From each of the 200 PlanetLab nodes used as destinations in this section, we issue both a traceroute and an RR ping to each of the 11 used as sources, so the RR ping will have the same source and destination as the traceroute (unlike with reverse traceroute's RR probes to intermediate routers). Since the RR slots may fill up before the probe reaches the destination, we only check if the traceroute matches the portion of the path that appears in the RR. After alias resolution, the median fraction of RR hops seen in the corresponding traceroute is 0.67, with the other factors described in this section accounting for the differences. This fraction is 0.2 lower than that for reverse traceroute, showing the difficulty in matching RR hops to traceroute hops.

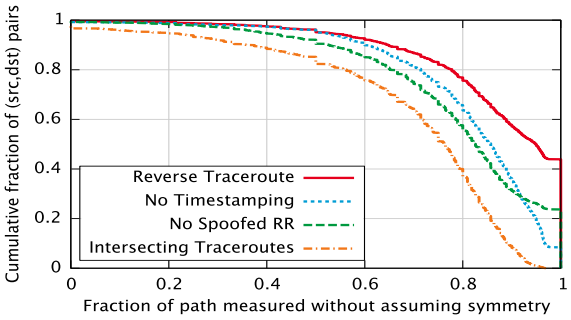


Figure 10: For the PlanetLab dataset, the fraction of reverse path hops measured, rather than assumed symmetric. The graph includes results with subsets of the reverse traceroute techniques.

5.2 Coverage

In Section 5.1, we noted that our paths are more accurate when our techniques succeed in measuring the entire path without having to fall back to assuming a link is symmetric. As seen in Figure 8, if forced to assume the entire path is symmetric, in the median case we would discover only 39% of the hops on a traceroute. In this section, we investigate how often our techniques are able to infer reverse hops, keeping us from reverting to assumptions of symmetry. Using the PlanetLab dataset, Figure 10 presents the results for our complete technique, as well as for various combinations of the components of our technique. The metric captured in the graph is the fraction of hops in the reverse traceroute that were measured, rather than assumed symmetric.

Reverse traceroute finds most hops without assuming symmetry. In the median path in the PlanetLab dataset, we measure 95% of hops (mean=87%), and in 80% of cases we are able to measure at least 78% of the path without assumptions of symmetric. By contrast, the traceroute intersection estimation technique from Figure 8 assumes in the median that the last 25% of the

path is symmetric (mean=32%). Although not shown in the graph, the results are similar for the traceroute server dataset – in the median case, reverse traceroute measures 95% of hops (mean=92%) without assuming symmetry.

The graph also depicts the performance of our technique if we do not use spoofed record route pings or do not use timestamping. In both cases, the performance drops off somewhat without both probing methods.

5.3 Overhead

We assess the overhead of our technique using the traceroute server dataset from Section 5.1, comparing the time and number of probes required by our system to those required by traceroute. The median (mean) time for one of the 10 PlanetLab sites to issue a traceroute to one of the 186 traceroute servers was 5 seconds (9.4 seconds). Using our current system, as available on `http://revtr.cs.washington.edu` and described in Section 4, the median (mean) time to measure a reverse traceroute was 41 seconds (116.0 seconds), including the time to send an initial forward traceroute (to determine if the destination is reachable and to present a round-trip path at the end). We have not yet pursued improving this aspect of the system. In the future, we will investigate lowering this delay by setting more aggressive timeouts for flaky PlanetLab vantage points, by cutting down on the communication overhead, and by attempting to adapt our probing rate to the rate limit of the particular target.

For each reverse traceroute measurement, our system sends the initial forward traceroute and a number of options-enabled ping packets, some of which may be spoofed. In cases when it is unable to determine the next reverse hop, it sends a forward traceroute and assumes the last hop is symmetric. In addition, we require traceroutes to build an atlas of paths to the source, and we use ongoing background mapping to identify adjacencies and to determine which vantage points are within RR-range of which prefixes.

If we ignore the probing overhead of the traceroute atlas and the mapping, in the median case, the only traceroute required is the initial one (mean=1.2 traceroutes). In the median (mean) case, a reverse traceroute requires 2 (2.6) record route packets, plus an additional 9 (21.2) spoofed RR packets. The median (mean) number of non-spoofed timestamp packets is 0 (5.1), and the median (mean) number of spoofed timestamp packets is also 0 (6.5). The median (mean) total number of options packets sent is 13 (35.4). As a point of comparison, traceroute uses around 45 probe packets on average, 3 for each of around 15 hops. At the end of a reverse traceroute, we also send 3 pings to each hop to measure latency. So, ignoring the creation of the various atlases, reverse traceroute generally requires roughly 2-3x more packets than traceroute.

IP address	AS name	Location	RTT
132.170.3.1	UCF	Orlando, FL	0ms
198.32.155.89	FloridaNet	Orlando, FL	0ms
198.32.132.64	FloridaNet	Jacksonville, FL	3ms
198.32.132.19	Cox Comm.	Atlanta, GA	9ms
68.1.0.221	Cox Comm.	Ashburn, VA	116ms
216.52.127.8	Internap	Washington, DC	35ms
66.79.151.129	Internap	Washington, DC	26ms
66.79.146.202	Internap	Washington, DC	24ms
66.79.146.241	Internap	Miami, FL	53ms
66.79.146.129	Internap	Seattle, WA	149ms

Table 1: Traceroute giving forward path from University of Central Florida to *66.79.146.129*.

The atlases represent the majority of our probe overhead. However, in many circumstances these atlases can be reused and/or optimized for performance. For example, if the source requests reverse paths for multiple destinations within a short period of time [45], we can reuse the atlas. As an optimization, we may need to only issue those traceroutes that are likely to intersect [22], and we can use known techniques to reduce the number of probes to generate the atlas [11]. We borrow the adjacency information needed for our timestamp probes from an existing mapping service [22]. To determine which spoofing vantage points are likely within record route range of a destination, we regularly issue probes from every spoofer to a set of addresses in each prefix. In the future, we plan to investigate if we can reduce this overhead by probing only a single address within each prefix.

6 Applications of Reverse Traceroute

We believe many opportunities exist for improving systems and studies using reverse traceroute. We next discuss three such examples of how reverse traceroute can be used in practice. We intend these sections to illustrate a few ways in which one can apply our tool; they are not complete studies of the problems.

6.1 Case study of debugging path inflation

Large content providers attempt to optimize client performance by replicating their content across a geographically distributed set of servers. A client is then redirected to the server to which it has minimum latency. Though this improves the performance perceived by clients, it can still leave room for improvement. Internet routes are often inflated [37], which can lead to round-trip times from a client to its nearest server being much higher than what they should be given the server's proximity. Using Google as an example, 20% of client prefixes experience more than 50ms latency over the minimum latency to the prefix's geographical region. Google wants a way to identify which AS is the cause of inflation, but it is hindered by the lack of information about reverse paths back to their servers from clients [21].

IP address	AS name	Location	RTT
66.79.146.129	Internap	Seattle, WA	148ms
66.79.146.225	Internap	Seattle, WA	141ms
137.164.130.66	TransitRail	Los Angeles, CA	118ms
137.164.129.15	TransitRail	Los Angeles, CA	118ms
137.164.129.34	TransitRail	Palo Alto, CA	109ms
137.164.129.2	TransitRail	Seattle, WA	92ms
137.164.129.11	TransitRail	Chicago, IL	41ms
137.164.131.165	TransitRail	Ashburn, VA	23ms
132.170.3.1	UCF	Orlando, FL	0ms
132.170.3.33	UCF	Orlando, FL	0ms

Table 2: Reverse traceroute giving reverse path from *66.79.146.129* back to University of Central Florida. The circuitous reverse path explains the huge RTT jump between the last two hops on the forward path. The third hop, 137.164.130.66 (internap-peer.lsanca01.transitrail.net), is a peering point between Internap and TransitRail in L.A.

As an illustration, we used reverse traceroute to diagnose an example of path inflation. We measured the RTT on the path from the PlanetLab node at the University of Central Florida to the IP address *66.79.146.129*, which is in Seattle, to be 149ms. Table 1 shows the forward path returned by traceroute, annotated with the locations of intermediate hops inferred from their DNS names. The path has some circuitousness going from Orlando to Washington via Ashburn and then returning to Miami. But, that does not explain the steep rise in RTT from 53ms to 149ms on the last segment of the path, because a hop from Miami to Seattle is expected to only add 70ms to the RTT[5].

To investigate the presence of reverse path inflation back from the destination, we determined the reverse path using reverse traceroute. Table 2 illustrates the reverse path, which is noticeably circuitous. Starting from Seattle, the path goes through Los Angeles and Palo Alto, and then returns to Seattle before reaching the destination via Chicago and Ashburn. We verified with a traceroute from a PlanetLab machine at the University of Washington that TransitRail and Internap connect in Seattle, suggesting that the inflation is due to a routing misconfiguration. Private communication with an operator at one of the networks confirmed that the detour through Los Angeles was unintentional. Without the insight into the reverse path provided by reverse traceroute, such investigations would not be possible by the organizations most affected by inflated routes.

6.2 Topology discovery

Studies of Internet topology rely on the set of available vantage points and data collection points. With a limited number available, routing policies bias what researchers measure. As an example, with traceroute alone, topology discovery is limited to measuring forward paths from a few hundred vantage points to each other and to other destinations. Reverse traceroute allows us to expose many peer-to-peer links invisible to traceroute.

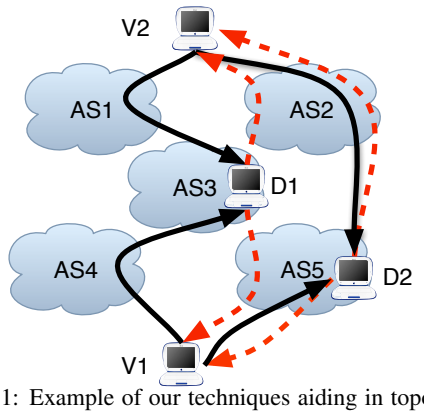


Figure 11: Example of our techniques aiding in topology discovery. With traceroutes alone, V1 and V2 can measure only the forward (solid) paths. If V2 is within 8 hops of D1, a record route ping allows it to measure the link AS3-AS2, and a record route ping spoofed as V1 allows it to measure AS3-AS5.

Figure 11 illustrates one way in which our techniques can uncover links. Assume that *AS3* has a peer-to-peer business relationship with the other ASes. Because an AS does not want to provide free transit, most routes will traverse at most one peer-to-peer link. In this example, traffic will traverse one of *AS3*'s peer links only if it is sourced or destined from/to *AS3*. *V1*'s path to *AS3* goes through *AS4*, and *V2*'s path *AS3* goes through *AS1*. Topology-gathering systems that rely on traceroute alone [22, 1, 34] will observe the links *AS1-AS3*, *AS4-AS3*, and *AS2-AS5*. But, they will never traverse *AS3-AS5*, or *AS3-AS2*, no matter what destinations they probe (even ones not depicted). *V2* can never traverse *AS1-AS3-AS5* in a forward path (assuming standard export policies), because that would traverse two peer-to-peer links. However, if *V2* is within 8 hops of *D1*, then it can issue a record-route ping that will reveal *AS3-AS2*, and a spoofed record route (spoofed as *V1*) to reveal *AS3-AS5* [6].

Furthermore, even services like RouteViews [27] and RIS [33], with BGP feeds from many ASes, likely miss these links. Typical export policies mean that only routers in an AS or its customers see the AS's peer-to-peer links. Since RouteViews has vantage points in only a small percentage of the ASes lower in the AS hierarchy, it does not see most peer links [29, 16].

To demonstrate how reverse traceroute can aid in topology mapping, we apply it to a recent study on mapping Internet exchange points (IXPs) [3]. That study used existing measurements, novel techniques, and thousands of traceroute servers to provide IXP peering matrices that were as complete as possible. As part of the

[5]Interestingly, the latency to Ashburn seems to also be inflated on the reverse path.

[6]Note that, because we only query for hops already known to be adjacent, our timestamp pings are not useful for topology discovery.

study, the researchers published the list of ASes they found to be peering at IXPs, the IXPs at which they peered, and the IP addresses they used in those peerings.

We measured the reverse paths back from those IP addresses to all PlanetLab sites. We discovered 9096 IXP peerings (triples of the two ASes and the IXP at which they peer) that are not in the published dataset, adding an additional 16% to the 58,534 peerings in their study. As one example, we increased the number of peerings found at the large London LINX exchange by 19%. If we consider just the ASes observed peering and not which IXP they were seen at, we found an additional 5057 AS links not in the 51,832 known IXP AS links, an increase of 10%. Of these AS links, 1910 do not appear in either traceroute [22] or BGP [40] topologies – besides not being known as IXP links, we are discovering links not seen in some of the most complete topologies available. Further, of the links in both our data and UCLA's BGP topology, UCLA classifies 1596 as Customer-to-Provider links, whereas the fact that we observed them at IXPs strongly suggests they are Peer-to-Peer links. Although the recent IXP study was by far the most exhaustive yet, reverse traceroute provides a way to observe even more of the topology.

6.3 Measuring one-way link latency

In addition to measuring a path, traceroute measures a round-trip latency for each hop. Techniques for geolocation [42, 18], latency estimation [22], and ISP comparisons [25], among others, depend on link latency measurements obtained by subtracting the RTT to either endpoint, then halving the difference (possibly with a filter for obviously wrong values). This technique should yield fairly accurate values if routes traverse the link symmetrically. However, previous work found that 88-98% of paths are asymmetric [15] resulting in substantial errors in link latency estimates [39]. More generally, the inability to isolate individual links is a problem when using network tomography to infer missing data – tomography works best only when the links are traversed symmetrically or when one knows both the forward and reverse paths traversed by the packets [10, 6].

A few alternatives exist for estimating link latencies but none are satisfactory. Rocketfuel infers link weights used in routing decisions [23], which may or may not reflect latencies. The geographic locations of routers provide an estimate of link latency, but such information may be missing, wrong, or outdated, and latency does not always correspond closely to geographic distance [18].

In this section, we revisit the problem of estimating link latencies since we now have a tool that provides reverse path information to complement traceroute's forward path information. Given path asymmetry, the reverse paths from intermediate routers likely differ from the end-to-end traceroutes in both directions. Without reverse path information from the intermediate hops back to the hosts, we cannot know which links a round-trip latency includes. Measurements to endpoints and intermediate hops yield a large set of paths, which we simplify using IP address clustering [22]. We then generate a set of linear constraints: for any intermediate hop R observed from a source S, the sum of the link latencies on the path from S to R plus the sum of the link latencies on the path back from R must equal the round-trip latency measured between S and R. We then solve this set of constraints using least-squares minimization, and we also identify the bound and free variables in the solution. Bound variables are those sufficiently constrained for us to solve for the link latencies, and free variables are those that remain under constrained.

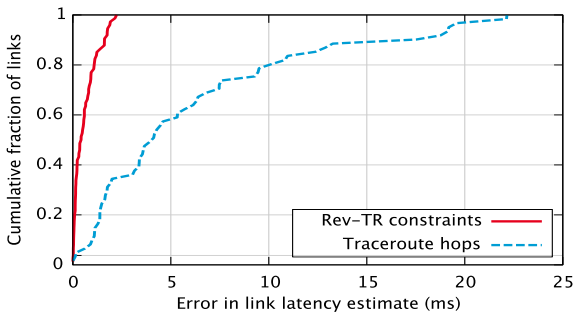


Figure 12: Error in estimating latencies for Sprint inter-PoP links. For each technique, we only include links for which it provided an estimate: 61 of 89 links using traceroute, and 74 of 89 using reverse traceroute. Ground truth reported only to 0.5ms granularity.

We evaluate our approach on the Sprint backbone network by comparing against inter-PoP latencies Sprint measures and publishes [38]. We consider only the directly connected PoPs and halve the published round-trip times to yield link latencies we use as ground truth, for 89 links between 42 PoPs. We observe 61 of the 89 links along forward traceroutes and 79 with reverse traceroute. We use these measurements to formulate constraints on the inter-PoP links, based on round-trip latencies measured from PlanetLab nodes to the PoPs using ping. This set of constraints allows us to solve for the latencies of 74 links, leaving 5 free and 10 unobserved.

As a comparison point, we use a traditional method for estimating link latency from traceroutes [22]. For each forward traceroute that traverses a particular Sprint link, we sample the link latency as half the difference between the round-trip delay to either end, then estimate the link latency to be the median of these samples across all traceroutes. Figure 12 shows the error in the latency estimates of the two techniques, compared to the published ground truth. Our approach infers link latencies with errors from 0ms to 2.2ms for the links, with a me-

dian of 0.4ms and a mean of 0.6ms. Because Sprint reports round-trip delays with millisecond granularity, the values we use for ground truth have 0.5ms granularity, so our median "error" is within the granularity of the data. The estimation errors using the traditional traceroute method range from 0ms to 22.2ms, with a median of 4.1ms and a mean of 6.2ms – $10x$ our worst-case, median, and mean errors. Based on this initial study of a single large network for which we have ground-truth, using reverse traceroute to generate and solve constraints yields values very close to the actual latencies, whereas the traditional approach does not.

7 Related work

Measurement techniques: Previous work concluded that too many paths dropped packets with IP options for options to form the basis of a system [13]. The Passenger and DisCarte projects, however, showed that the record route option, when set on traceroute packets, reduces false links, uncovers more routers, and provides more complete alias information [36, 35]. Hubble demonstrated the use of spoofed packets to probe a path in one direction without having to probe the other [19], but it does not determine the routers along the reverse path. Addressing this limitation in Hubble was part of the original motivation for this work.

The contributions of these various projects is in how they employ existing IP techniques – options and spoofing – towards useful ends. Our work employs the same IP techniques in new ways. We demonstrate how spoofing with options can expose reverse paths. Whereas Passenger and DisCarte used RR to improve forward path information, we use RR in non-TTL-limited packets to measure reverse paths. As far as we are aware, our work is the first to productively employ the timestamp option.

Techniques for inferring reverse path information: Various earlier techniques proposed methods for inferring limited reverse path information. Before such packets were routinely filtered, one study employed loose source-routing [31] to measure paths from numerous remote sites. Other interesting work used return TTL values to estimate reverse routing maps towards sources; however, the resulting maps contained less than half the actual links, as well as containing multiple paths from many locations [7]. PlanetSeer [43] and Hubble [19] included techniques for isolating failures to either the forward or reverse path; neither system, however, can give information about where on a reverse path the failure occurs. Netdiff inferred path asymmetry in cases where hop counts differ greatly in the two directions [25]; however, as our example in Section 6.1 shows, very asymmetric paths can have the same hop count. Tulip used ICMP timestamps (not the IP timestamp option we use) and other techniques to identify reordering and loss along either the forward or reverse path [24].

Systems that would benefit from reverse path information: Many systems seem well-designed to make use of reverse path information, but, lacking it, make various substitutions or compromises. We mention some recent ones here. Geolocation systems use delay and path information to constrain the position of targets [14, 18, 42], but, lacking reverse path data, are under constrained. iPlane shows that knowledge of a few traceroutes from a prefix greatly improves path predictions [22], but lacks vantage points in most. iSpy attempted to detect prefix hijacks using forward-path traceroutes, yet the signature it looked for is based on the likely pattern of reverse paths [46]. Similarly, intriguing recent work on inferring topology through passive observation of traffic bases its technique on an implicit assumption that the hop counts of forward and reverse paths are likely to be the same [12]. Similarly, systems for network monitoring often assume path symmetry [8, 25]. All these efforts can potentially benefit from the work described here.

8 Conclusion

Although widely-used and popular, traceroute is fundamentally limited in that it cannot measure reverse paths. This limitation leaves network operators and researchers unable to answer important questions about Internet topology and performance. To solve this problem, we developed a reverse traceroute system to measure reverse paths from arbitrary destinations back to the user. The system uses a variety of methods to incrementally build a path back from the destination hop-by-hop, until it reaches a known baseline path. We believe that our system makes a strong argument for both the IP timestamp option and source spoofing as important measurement tools, and we hope that PlanetLab and ISPs will consider them valuable components of future measurement testbeds.

Our reverse traceroute system is both effective – in the median case finding all of the PoPs seen by a direct traceroute along the same path – and useful. The tool allows operators to conduct investigations impossible with existing tools, such as tracking down path inflation along a reverse route. Many operators seem to view reverse traceroute as a useful tool – based on the results presented in this paper, we received requests to help us test the tool and offers of spoofing vantage points, including hosts at all the PoPs of an international backbone network. The system's probing methods have also proved useful for topology mapping. In illustrative examples, we demonstrated how our system can discover more than a thousand peer-to-peer links invisible to both BGP route collectors and to traceroute-based mapping

efforts, as well as how it can be used to accurately measure the latency of backbone links. We believe the accuracy and coverage of the tool will only improve as we add additional vantage points. A demo of our tool is available at `http://revtr.cs.washington.edu`.

Acknowledgments

We gratefully acknowledge John Byers, our shepherd, and the anonymous NSDI reviewers for their valuable feedback on earlier versions of this paper. Adam Bender, Rob Sherwood, Ricardo Oliveira, Brice Augustin, and Balachander Krishnamurthy provided access to and help with their tools and data. This work was funded partially by Google, Cisco, and NSF grants CNS-0905568 and MRI-0619836. We are thankful for their support.

References

[1] Archipelago measurement infrastructure. `http://www.caida.org/projects/ark/`.

[2] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *IMC*, 2006.

[3] B. Augustin, B. Krishnamurthy, and W. Willinger. IXPs: Mapped? In *IMC*, 2009.

[4] A. Bender, R. Sherwood, and N. Spring. Fixing Ally's growing pains with velocity modeling. In *IMC*, 2008.

[5] R. Beverly, A. Berger, Y. Hyun, , and k claffy. Understanding the efficacy of deployed Internet source address validation filtering. In *IMC*, 2009.

[6] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. In *SIGMETRICS*, 2002.

[7] H. Burch. *Measuring an IP network in situ*. PhD thesis, CMU, 2005.

[8] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM*, 2004.

[9] D. Choffnes and F. E. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *SIGCOMM*, 2008.

[10] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, 2002.

[11] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *SIGMETRICS*, 2005.

[12] B. Eriksson, P. Barford, and R. Nowak. Network discovery from passive measurements. In *SIGCOMM*, 2008.

[13] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica. IP options are not an option. Technical report, EECS Department, University of California, Berkeley, 2005.

[14] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation of Internet hosts. *IEEE/ACM TON*, 2006.

[15] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the Internet. In *Autonomic Networks Symposium in Globecom*, 2005.

[16] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. A systematic framework for unearthing the missing links. In *NSDI*, 2007.

[17] Internet address hitlist dataset, PREDICT ID USC LANDER internet_address_hitlist_it28wbeta20090914. `http://www.isi.edu/ant/lander`.

[18] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *IMC*, 2006.

[19] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying black holes in the Internet with Hubble. In *NSDI*, 2008.

[20] K. Keys, Y. Hyun, and M. Luckie. Internet-scale alias resolution with MIDAR. Under preparation, 2010.

[21] 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 *IMC*, 2009.

[22] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.

[23] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *IMW*, 2002.

[24] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet path diagnosis. In *SOSP*, 2003.

[25] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *NSDI*, 2008.

[26] Measurement Lab website. `http://www.measurementlab.net/`.

[27] D. Meyer. RouteViews. `http://www.routeviews.org`.

[28] NANOG 45, 2009.

[29] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. In search of the elusive ground truth: The Internet's AS-level connectivity structure. In *SIGMETRICS*, 2008.

[30] Outages mailing list. `http://isotf.org/mailman/listinfo/outages`.

[31] J.-J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *SIGCOMM CCR*, 1998.

[32] RIPE 58, 2009.

[33] RIPE RIS. `http://www.ripe.net/ris/`.

[34] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. *SIGCOMM CCR*, 2005.

[35] R. Sherwood, A. Bender, and N. Spring. Discarte: A disjunctive internet cartographer. In *SIGCOMM*, 2008.

[36] R. Sherwood and N. Spring. Touring the Internet in a TCP sidecar. In *IMC*, 2006.

[37] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *SIGCOMM*, 2003.

[38] Sprint IP network performance. `https://www.sprint.net/performance/`.

[39] R. A. Steenbergen. A practical guide to (correctly) troubleshooting with traceroute. In *NANOG 45*, 2009. `http://www.nanog.org/meetings/nanog45/presentations/Sunday/RAS_traceroute_N45.pdf`.

[40] UCLA Internet topology collection. `http://irl.cs.ucla.edu/topology/`.

[41] `http://www.merit.edu/mail.archives/nanog/`. North American Network Operators Group mailing list.

[42] B. Wong, I. Stoyanov, and E. G. Sirer. Octant: A comprehensive framework for the geolocalization of Internet hosts. In *NSDI*, 2007.

[43] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *OSDI*, 2004.

[44] Y. Zhang, Z. M. Mao, and M. Zhang. Effective diagnosis of routing disruptions from end systems. In *NSDI*, 2008.

[45] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. *ACIRI Technical Report*, 2000.

[46] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSpy: detecting IP prefix hijacking on my own. In *SIGCOMM*, 2008.