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. Our approach employs multiple vantage points to measure a rich set of known paths towards the source, then uses these paths as a basis to allow us to extrapolate to unknown paths. We use a variety of measurement techniques to piece together the path from the destination incrementally until it intersects one of the known paths. 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 use our reverse traceroute system to study previously unmeasurable aspects of the Internet: we uncover more than a thousand peer-to-peer AS links invisible to current topology mapping efforts, we present a case study of how a content provider could use our tool to troubleshoot poor path performance, 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 [36]. Researchers use it as the basis for Internet maps [1, 21, 30], path prediction [21], geolocation [40, 13], ISP performance analysis [24], and anomaly detection [44, 19, 42, 41].

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 [14]. 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" [36].

This invisibility hinders operators. For instance, although Google has data centers distributed around the world, 1 in 5 client prefixes experience unreasonably high latency, even with a nearby server. A Google group trying to improve this performance concluded that, "in order to more precisely troubleshoot problems, [Google] needs the ability to gather information about the reverse path back from clients to Googles nodes" [20].

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 [21], geolocation [40, 13], ISP performance analysis [24], and prefix hijack detection [44]. Recent work shows that measured topologies miss many of the Internet's peer-to-peer links [26, 15] because mapping projects [1, 21, 30] 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 [27, 39]. 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 [30, 8], 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 infor-

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

mation 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 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 components of the Internet.

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 infer information unavailable from any single one. We first measure 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 [32, 31, 12], while remaining perfectly safe, as the spoofed source address is always 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 [31], 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 traceroute has numerous limitations and caveats, and can be potentially misleading, yet remains an extremely useful tool [36]. Similarly, we want users and readers to recognize the limitations and caveats of our tool, while simultaneously realizing its promise and potential uses. Towards this end, Section 5.1 includes a thorough discussion of how the output of our tool might differ from a traceroute along the same path, 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 destination points to a limitation of our tool compared to traceroute; whereas traceroute can measure most of the path towards an unresponsive or unreachable destination, our tool make a best-guess assumption that it is measuring the proper path.

We rely on routers "friendly" to our techniques, some of which have the potential for abuse and can be tricky for novices to use without causing disturbances, and we ultimately want our tool widely used operationally. We have attempted to pursue our approach in a way that encourages continued router support. We presented the work early on at operators' conferences [16, 17], and so far the response has been overwhelming 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. Our system performs measurements in a way that emphasizes network friendliness, controlling probe rate across all measurements.

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 at the PoP-level. We use our reverse traceroute tool to measure link latencies in the Sprint backbone network with less than a millisecond of error, on average. 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.

2 Background

In this section, we provide 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. Between two ASes, different peering points may be used in the two directions due to policies such as early-exit/hot-potato routing. Even within an individual AS, different paths may be chosen due to traffic engineering objectives.

Standard traceroute tool: Traceroute is used to measure the sequence of routers from a source to a destination, without requiring control of the destination. At the time 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 time exceeded responses from each of the routers on the forward 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 on the path.

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]. We use the Paris option that measures a single consistent path for all our traceroutes. 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 proven 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 then coordinates probes from the source and 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, 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)). In Section 3.3, we provide an example of combining measurement techniques and the path atlas to measure a complete path.

3.1 Identify Reverse Hops with IP Options

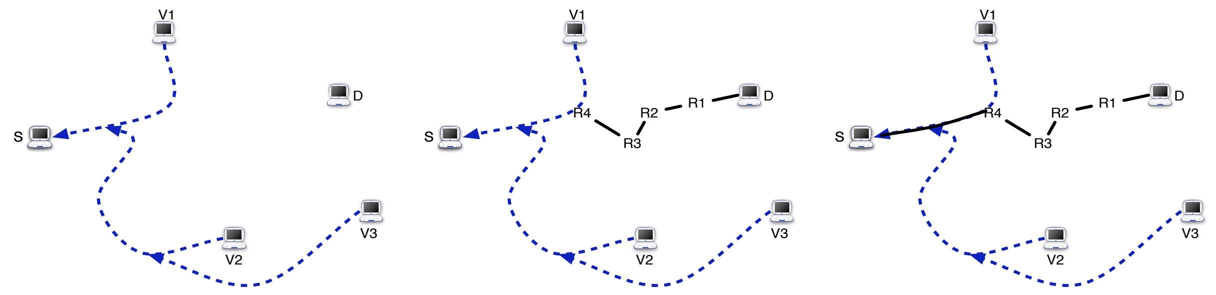
Two basic measurement primitives we use are the Record Route and Timestamp IP options. While TTL values are reset by the destination, restricting traceroute to measuring only on the forward path, IP options are generally reflected in the reply from a destination, so routers along both the forward and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

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

IP timestamp option (*TS*): IP allows ICMP probes to query the timestamps of a set of specific routers along a path. 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 in the following two probes to gather reverse hops:

- *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

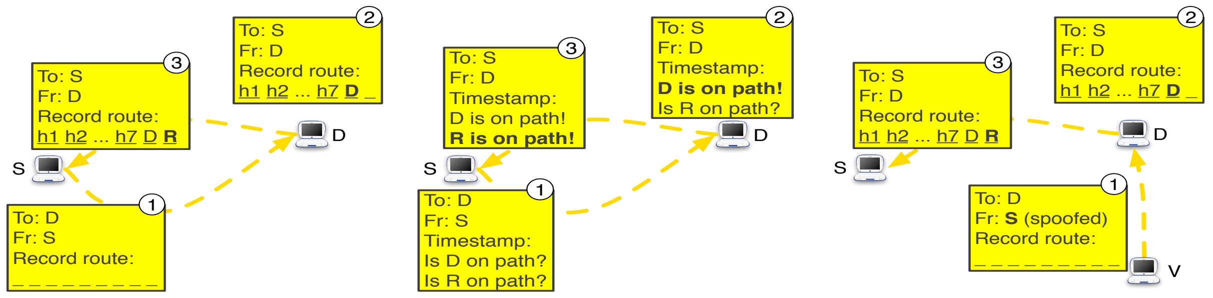


(a) Vantage points traceroute to S, creating an atlas of known paths.

(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.

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 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 or refute our hypothesis.

We use existing network topology information – specifically IP-level connectivity of routers from Internet mapping efforts [21] – 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 probe packet to one other than its own. We use a limited form of spoofing, where the only IP addresses we spoof are those of our vantage points. This form of spoofing is an extremely powerful measurement tool. When V_1 probes D spoofing as V_2, D's response will go to V_2; we refer to V_1 as the spoofer and V_2 as the receiver. This method allows the probe to traverse the path from D to V_2 without having to traverse the path from V_2 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_1 to V_2, but source routed through D) [28]. 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 meaurements to all routable prefixes, many routers that filter source route option-enabled packets do not filter timestamp and 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 measure we want to capture. Hubble used limited spoofing to check one-way reachability [19]; we use it 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 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 is not anomalous: the Spoofer project estimates that 20% of IP addresses can send spoofed packets [33]. 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 in most cases. 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.

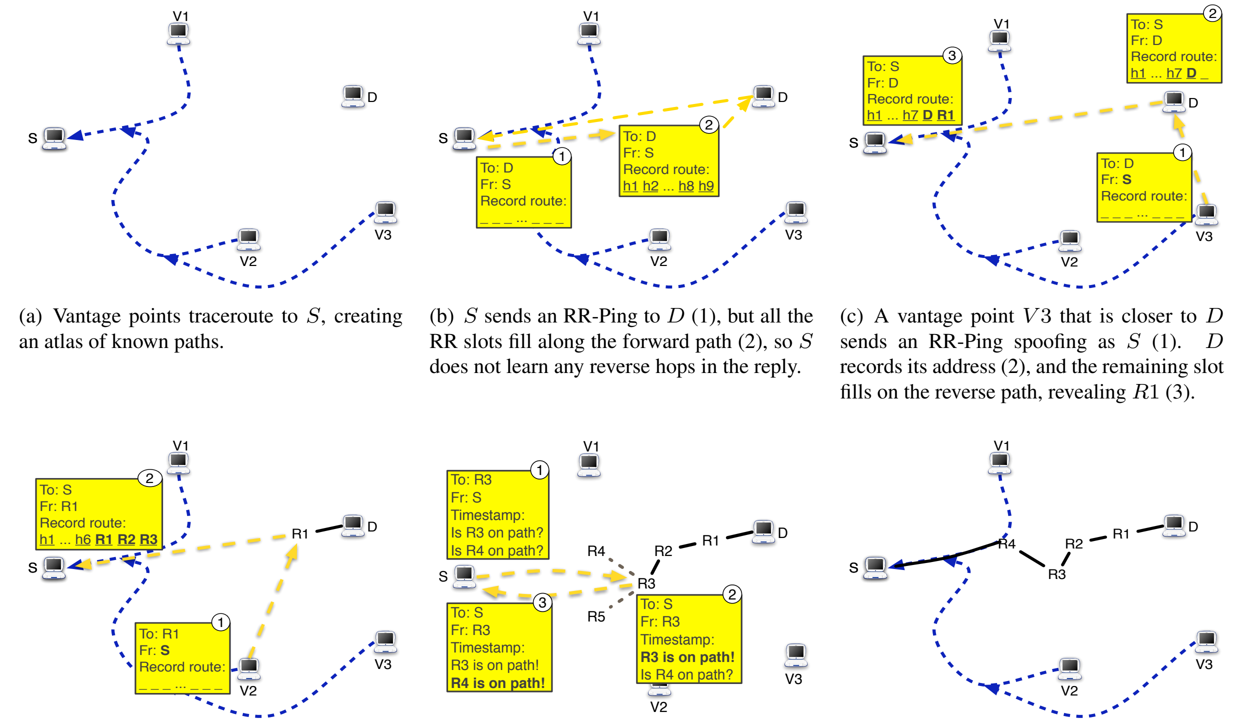
Figure 3 illustrates how we can compose the above set of techniques to determine the reverse path from D to S, when we have control over S and a set of other vantage points (V_1, V_2, V_3). We assume that we have network topology information that provides a partial map of router-level connectivity, e.g.,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

We begin by having the vantage points issue traceroute probes to S (Figure 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*$(S \to D)$ 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 (§4.4 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 $V_3 \to D \to S$ 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)). 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

[1]This is not quite as simple as sending a TTL=8 limited probe, because of issues with record route implementations [31]. 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 depicting the reverse traceroute technique.

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 (Figure 3(f)). Once the procedure has determined the hops in the reverse path, we issue pings from the source to each one of the hops 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 §5.1 and §5.2, we present results that we usually do not have to assume many symmetric hops and that we still achieve highly accurate paths.

4 System Implementation

Section 3.3 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, we require our system design to be driven by the following requirements.

- *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.
- *Accuracy:* It should be robust to variations in how options-enabled packets are handled by routers.

4.1 Architecture

Our system consists of vantage points (VPs), which issue measurements, a controller, which coordinates the

[2]Measurement techniques may discover different addresses on a router [32], so we determine intersections using alias data from topology mapping projects [1, 21, 31] and a state-of-the-art technique [4].

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. 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 [21][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 iPlane 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. We use alias information from existing projects [21, 31] to identify IP addresses collocated on a single router.

4.2 Current Deployment

In our current deployment, we employ 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 use 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. Various organizations provide public web-accessible traceroute servers, and we use 500 of them in our system. 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, so 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 of their choice. 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 wider launch that does not perform up to expectations.

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

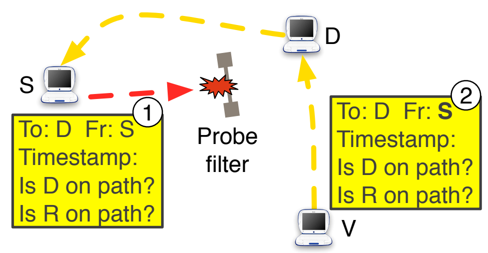


Figure 4: If a timestamp probe from S encounters a filter (1), we can often bypass it by spoofing as S from a different vantage point (2), as long as the filter is just on the forward path.

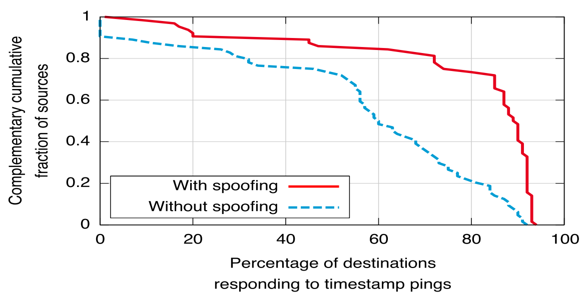


Figure 5: For 100 random destinations and 100 PlanetLab nodes, fraction of destinations from which each PlanetLab node receives responses to timestamp pings. We compare the fraction when the node sends the pings directly to the fraction when 5 spoofers spoof as the node.

4.3 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 4, 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 100 destinations selected at random from those we know respond to timestamp probes from at least one VP. We choose 100 random non-spoofing PlanetLab nodes and the 5 spoofing PlanetLab nodes we found to receive (non-spoofed) timestamp responses from the highest number of destinations. First, each non-spoofing node sends a timestamp ping to each destination, then each spoofing node sends 100 pings to each destination, spoofing as each non-spoofing node in turn. In the non-spoofed case, we send redundant probes to account for loss caused by something other than permanent filters. Figure 5 shows the distribution of how many destinations each source receives responses from, both without and with spoofing. Our results show that spoofed times-

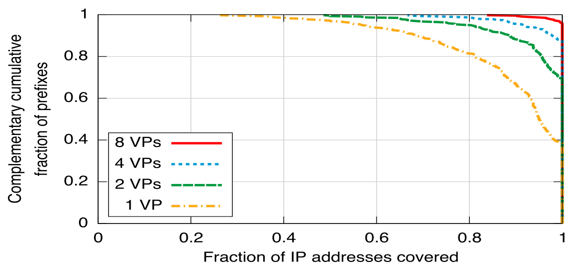


Figure 6: For prefixes in which iPlane observed $\geq$ 15 addresses, the fraction of the addresses for which we can find reverse hops using RR probes from a given number of vantage points per prefix. Note that we only include addresses within range of at least one vantage point. Prefixes with few addresses 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.

tamp probes allow us to avoid many of the forward path filters and successfully probe along the reverse path. In reverse traceroute's timestamp measurements, we use exactly this approach by retrying with 5 spoofers whenever the source does not receive a response. Since a small number of vantage points have filter-free paths to most responsive destinations, we use the 5 best overall, rather than choosing per destination.

4.4 Selective Use of Vantage Points for Scalability

With spoofed RR, only nearby VPs will end up recording reverse hops, since space in the options area of IP packets is limited. In order to quickly discover reverse hops, avoid rate limiting, and reduce overhead, our goal is to determine which VPs are likely 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 each VP issues a record route ping to every address. 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 VPs 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 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 6 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.

The results show that a small number of VPs suffices to find reverse hops for most training addresses within most prefixes. We will encounter previously unseen addresses in the course of reverse traceroutes, and so we evaluate how well these orderings work for novel addresses. To do so, we consider the DisCarte topology [31] and a day's iPlane topology. To evaluate our per-prefix orders, we train the set cover algorithm on measurements from our VPs to all DisCarte addresses. We then consider the 7823 responsive iPlane destinations which do not appear in the DisCarte topology, but which are in prefixes seen in DisCarte. For each destination, we use the DisCarte-derived set cover ordering to issue one record route probe at a time until we discovered a reverse hop. For 80% of destinations we discover a reverse hop after only a single probe, for more than 90% we need at most 2 VPs, and we receive very little benefit from using more than 5 VPs.

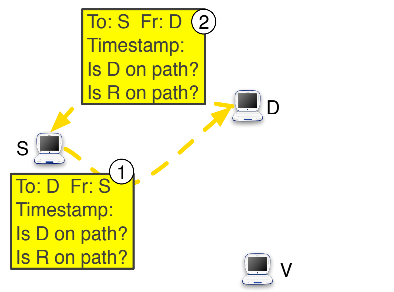
4.5 Correcting for Variations in Router Support

We next explain how we compensate for variations in router 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 19.4% of IP 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 7 illustrates our fallback technique, as described in the caption. 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.

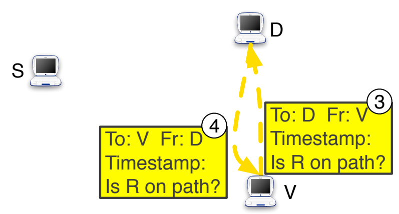
5 Evaluation

5.1 Accuracy

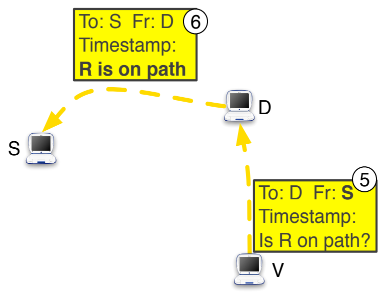
As an initial test of how well our reverse traceroute system can determine reverse paths, we consider evaluation settings that allow us to compare a reverse traceroute from D to a S to a direct traceroute from D to S. A complete evaluation of the accuracy of our technique would require ground truth information 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



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

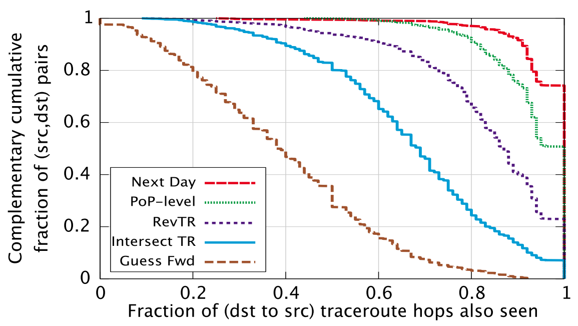


Figure 8: For reverse traceroute and techniques for approximating the reverse path, the fraction of hops on a direct traceroute from the destination to the source that the technique also discovers. Dataset includes paths from 200 PlanetLab destinations back to 11 PlanetLab sources. [The key labels are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

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 hosts at 11 PlanetLab sites chosen at random from the spoofing nodes. As destinations, we use one host at each of the 200 PlanetLab sites that do not allow spoofing. Although such a set cannot be representative of the entire Internet, the set of destinations includes hosts in 35 countries. The measured reverse paths traversed 13 of the 14 transit-free commercial networks, including all 8 Tier 1 ISPs, and 8 of 13 non-transit-free large Tier 2 networks [37]. Previous work observed load balancing in many of these large networks [2], providing a good test for our techniques.

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. 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 [1, 21, 31] 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 [32]. 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. Despite this drawback, we believe that using alias information leads to the most meaningful comparison we can present. First, a comparison that did not consider aliases would be misleading; in fact, it would generally consider different even the hops seen on a traceroute and a record route from the same source to the same destination [31]. Second, we rely on current techniques designed to reduce false postives. Third, based on analysis we discuss later in this section, we believe that missing aliases distort our results more than false aliases. 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.

The figure also compares 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* line depicts 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. Another approach would be to measure traceroutes from a set of vantage points to the source. Using iPlane's PlanetLab and traceroute server measurements, the *Intersect TR* line 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. 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.

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. Recent work on this technique found that only 31% of addresses seen on traceroutes between PlanetLab nodes respond in a way that allows this determination [4]. 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 alias resolution. 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.

In these cases, it is possible or even likely that the two measurement techniques observe IP addresses that are actually 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 [21]. iPlane has PoP mappings for 71% of the missing addresses and 77% of the extra ones. Figure 8 includes a *PoP-Level* line 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 (mean=94%).

For many applications such as diagnosis of inflated latencies, PoP level granularity suffices. As a second study, to demonstrate the performance of our system on paths that originate outside PlanetLab, we use public traceroute servers. We again use 11 PlanetLab sites as sources, and we issue traceroutes towards each site from a hundred random traceroute servers. Because in many cases we do not know the IP address of the TR server, we use the first hop along its path as the destination in our reverse traceroute measurements. Because existing sets of alias data rely largely on measurements within PlanetLab [21, 31, 4], we omit alias-level comparisons as being misleading due to the sparseness of the clustering data. In the median case, a reverse traceroute along the path from a traceroute server back to a PlanetLab host sees all of the PoPs seen on the traceroute issued directly from the traceroute server (mean=94%). In 89% of cases, reverse traceroute returns hops in the PoPs of at least 80% of the traceroute hops.

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* line in Figure 8 compares how many of the traceroutes' hops are also on traceroutes issued the following day. In 26% of cases, the traceroutes differ. In a loose sense, this result suggests 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 3% of routers and 16% of IP addresses appear with RR but do not appear in traceroutes [32, 31]. *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 they are load-balanced per-packet, rather than per-flow [31].

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 tun-

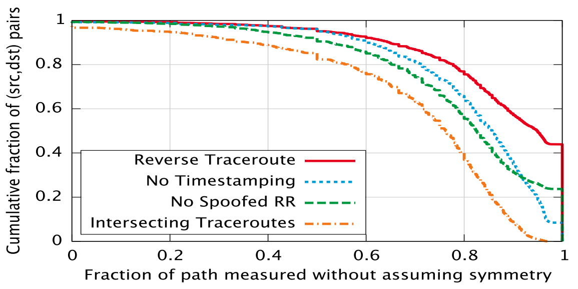


Figure 9: The fraction of reverse path hops meausured, rather than assumed symmetric (the fallback). Graph also includes results when only employing some of the reverse traceroute techniques. Dataset includes paths from 200 PlanetLab destinations back to 11 PlanetLab sources.

nels, 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 destination as the TR (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 TR hops and suggesting that our assumption of destination-based routing causes few of the differences between reverse traceroute and traceroute hops.

5.2 Coverage

In Section 5.1, we discussed the similarity between our path and a direct traceroute, and reasons why they might differ from each other or from the actual path. An additional reason is that reverse traceroute falls back on assuming a link is symmetric when our techniques fail to measure a reverse hop. 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 9 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, 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%).

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 estimate the overhead of our technique using the datasets from Section 5.1, comparing the time and number of probes required by our system compared to traceroute. It took the 11 PlanetLab sites an average of 9.3 seconds to issue a traceroute to one of the 100 traceroute servers. Using our current system, as available on `http://revtr.cs.washington.edu` and described in Section4, it took an average of 15.6 seconds to measure the reverse traceroute.

The system is not currently instrumented to track the number of probes. Instead, we describe the overhead of our initial implementation, used to measure reverse paths from 200 PlanetLab destinations back to 11 sources. These measurements were intended to demonstrate the feasibility of our technique, and so predate the implementation described in Section 4. We describe a few differences that increase the overhead versus that incurred by the live system. First, rather than smartly determining the set of vantage points likely to be within 8 hops of a target, then spoofing as one of them, we simply sent spoofed record route probes from each site, as well as a non-spoofed RR from the source. When the source received responses to these, it selected the one that gives the most reverse hops. Second, rather than guessing adjacent routers and testing in series with timestamp until one is verified on the reverse path, we sent out the probes in a batch. When the source received the responses, it selected whichever confirmed a reverse hop. Both these details amounted to parallelizing the measurements in a way that required sending extra probes, but simplified the process. Because we were measuring paths for all 200 destinations in parallel and randomizing the probing order per VP, we did not worry about rate limiting. So when we found a hop on the reverse path that did not have a path to the source in our traceroute atlas, we sent a set of 44 record route packets (one from the source and one from each of the 43 spoofing sites available at the time). If the RR probes did not discover a reverse hop, we sent a set of timestamp packets, one per adjacent router.

The median (mean) probe overhead required by reverse traceroute for a (src,dst) pair are 5 (5.7) RR sets and 4 (4.8) TS sets. The 90th percentiles are 11 RR and 10 TS sets of probes. The median (mean) number of record route packets and timestamp packets required are 220 (251.5) and 75 (146.7). The 90th percentile number of record route packets is 484, and the 90th percentile number of timestamp packets is 379. As a point of comparison, traceroute uses around 45 probe packets on average, 3 for each of around 15 hops. So, on average, a reverse traceroute requires the equivalent of 5.6 traceroutes of RR packets and the equivalent of 3.3 traceroutes in TS packets.

In addition, we issue a standard traceroute towards the destination, and a traceroute from each vantage point towards the source to build our atlas of paths. These initial traceroutes represent the majority of our probe overhead. However, we issue the traceroutes in parallel, and each vantage point issues only a single traceroute. If the source requires reverse paths from multiple destinations within a period over which the paths are likely stable [43], we can reuse this atlas and amortize the costs of those probes. In the future, we will explore whether we can get equivalent results with fewer vantage points, as well as use existing techniques to reduce the number of probes to generate the atlas [10].

6 Applications of Reverse Traceroute

We next discuss how reverse traceroute can be used, starting with two examples of recent studies that would benefit from our system.

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 [34], 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 point to an AS as the cause of inflation, but is hindered by the lack of information about reverse paths back to their servers from clients [20].

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.[4]

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*.

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 Los Angeles.

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 op-

[4]Interestingly, the latency to Ashburn is also inflated. This is likely also due to reverse path inflation given the direct forward path to Ashburn.

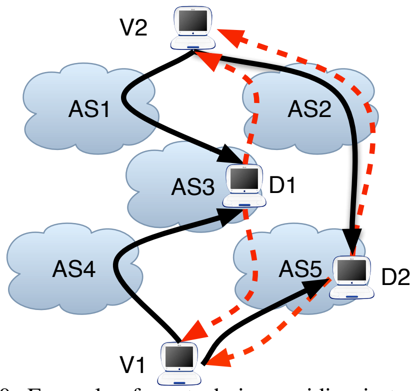


Figure 10: Example of our techniques aiding in topology discovery. With traceroutes alone, V1 and V2 can measure only the forward (solid) paths. If V2 is within 8 hops of D1, a record route ping allows it to measure the link AS3-AS2, and a record route ping spoofed as V1 allows it to measure the link AS3-AS5.

erator 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 can 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 of the peer-to-peer links invisible to traceroute.

Figure 10 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 [21, 1, 30] 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 path segment traverses 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* [5].

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

Furthermore, even services like RouteViews [25] and RIS [29], 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 [26, 15].

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 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 [21] or BGP [38] 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 [40, 18], latency estimation [21], and ISP comparisons [24], 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 [14] resulting in substantial errors in link latency estimates [36]. 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 symmetri-

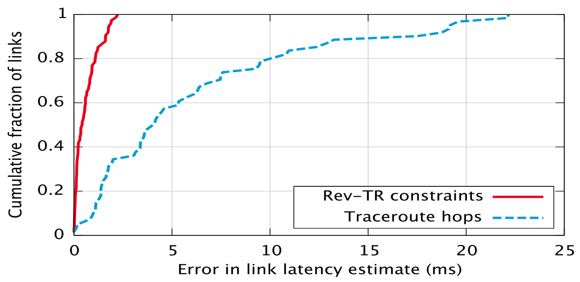


Figure 11: Error in estimating latencies for Sprint inter-PoP links. For each technique, we only include links for which it provided an estimate: 61 of 89 links using traceroute, and 74 of 89 using reverse traceroute. Using reverse traceroute to generate and solve constraints yields values very close to the actual latencies, whereas the traditional method does not.

cally or when one knows both the forward and reverse paths traversed by the packets [9, 5].

A few other alternatives exist for estimating link latencies but none are satisfactory. Rocketfuel annotates links with their link weights used in routing decisions [22], which may or may not reflect latencies. The geographic locations of routers provide an estimate of link latency, but geolocation 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 [21]. 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.

We evaluate our approach on the Sprint backbone network by comparing against inter-PoP latencies available on their website[35]. We consider only the directly connected PoPs and halve the published round-trip times to yield link latencies we use as ground truth, 89 links on 42 PoPs. Using traceroute and reverse traceroute, we observe 61 of the 89 links along forward traceroutes and 79 along reverse paths. 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 [21]. 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 11 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 median 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. Additionally, traceroute only observed, and hence could only provide estimates for, 61 of the links, whereas we provide estimates for 74. Based on this initial study of a single large network for which we have ground-truth, our reverse traceroute-based technique shows promise for more accurate latency estimation that may benefit a range of applications.

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 [12]. 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 [32, 31]. 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 [28] 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 [6]. PlanetSeer [41] and Hubble [19] gave 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 the forward hop count differs greatly from the reverse hop count [24]; 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 [23].

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 [13, 18, 40], but, lacking reverse path data, are under constrained. iPlane shows that knowledge of a few traceroutes from a prefix greatly improves path prediction accuracy [21], 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 [44]. 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 [11]. Similarly, systems for network monitoring often assume path symmetry [7, 24]. 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 the reverse traceroute system to measure reverse paths from arbitrary destinations back to the user. The system builds off a baseline atlas of paths from its distributed vantage points to the user, uses IP options to identify hops on the reverse path, exploits the Internet's destination-based routing to stitch together the hops, and employs source spoofing to best use the vantage points. 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.

The 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 system's probing methods prove extremely useful for topology mapping, discovering more than a thousand peer-to-peer links invisible to both BGP route collectors and to traceroute-based mapping efforts. By exposing these hidden links, we present an enriched view of the AS degree distribution. The tool also 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. We believe the accuracy and coverage of the tool will only improve as we add additional vantage points. A beta version of our tool is available for use 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 NSF, Google, and Cisco, and 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] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network Tomography on General Topologies. In *ACM SIGMETRICS*, 2002.

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

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

[8] 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.

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

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

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

[12] 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.

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

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

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

[16] E. Katz-Bassett. Practical reverse traceroute. In *NANOG 45*, 2009. http://www.nanog.org/meetings/nanog45/presentations/Tuesday/Katz_reversetraceroute_N45.pdf.

[17] E. Katz-Bassett. Reverse traceroute. In *To appear, RIPE 58*, 2009.

[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] 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.

[21] 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.

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

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

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

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

[26] 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.

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

[28] J.-J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *SIGCOMM CCR*, 28(1):41–50, January 1998.

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

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

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

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

[33] The Spoofer project. http://spoofer.csail.mit.edu/.

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

[35] Sprint.net looking glass. https://www.sprint.net/lg/.

[36] 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.

[37] Tier 1 networks Wikipedia. http://en.wikipedia.org/wiki/Tier_1_network.

[38] UCLA internet topology collection. http://irl.cs.ucla.edu/topology/.

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

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

[41] 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.

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

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

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

9 Appendix

9.1 Router Responsiveness and Rate-Limiting

Because we only care about measurements from nearby vantage points, we set the initial TTL to 11 for all RR and spoofed RR probes. Probes from distant vantage points will expire before reaching the destination, limiting the effects of rate-limiting. We use a value of 11 to account for the fact that record route and TTL hop counts may differ from each other [31].

9.2 Filtering and record route

If a timestamp probe from a source fails to receive a response, we can have a number of vantage points probe the destination at once, spoofing as the source. We need not worry about rate limiting–any probe that reaches the destination and receives a response is equivalent. However, at mentioned above, we cannot probe simultaneously with record route, as we need to make sure that probes from nearby vantage points are not rate limited. But, we would still like to differentiate between cases when the destination does not respond to these probes and cases when probes are being filtered (but might work over a different path). Since we can send only a few probes at a time, we would like to limit the number of sets we have to send before deciding that a destination is simply not responsive. To set an appropriate threshold, we probe every IP address in the DisCarte topology from every PlanetLab site, and we consider every site that received responses from at least 10,000 destinations (so ran properly and does not have a filter near the source). Figure 12 shows the fraction of those sites that received responses from the destinations. We see that, if a destination responds to a few vantage points, it most likely responds to many. In measuring a reverse path, we

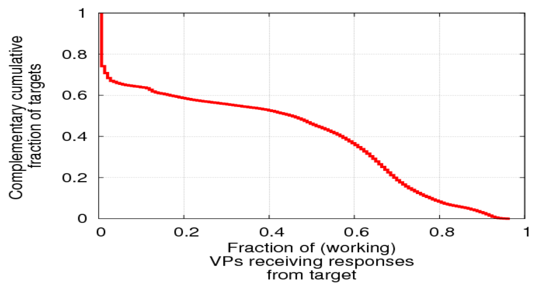


Figure 12: Fraction of PlanetLab sites receiving responses to record route pings sent to all addresses in the DisCarte topology.

first attempt a non-spoofed ping, then if it fails attempt spoofed probes from 3 vantage points at a time. We give up after 3 such sets if none yields a response. Making the simplifying assumption that each vantage point is equally likely to receive a response, we would expect to get a response with probability at least $0.89 = 1-(1-0.2)^{10}$ for the 59% of destinations that reply to at least 20% of sites and probability at least 0.99 for the 52% of destinations that reply to at least 40% of sites. We consider this adaquete and do not worry about the destinations that reply to fewer vantage points, since exhaustive search would take too long to justify the limited benefit.

9.3 Timestamp overstamping

We found that some hosts record stamps in two slots, even if we only ask for their timestamp once, and even if the second address we ask for is provably not on the path. All Linux machines we tested display this bug. In the DisCarte topology, 5.4% of addresses display this behavior. If we fail to recognize these cases, then any address we guess as being adjacent will appear to have stamped and be declared as being on the reverse path. We account for this over-stamping by testing for R on the reverse path by asking for timestamps from D, R, R; since we have never observed any hosts over-stamping two slots, if the third slot is stamped then R must have done it. However, many routers will only stamp once, even if asked twice, so, if we only receive two stamps, we need to decide whether R made the second stamp or D over-stamped into it. We initially thought to test by seeing if the value of the first and second stamps were the same. However, by testing with addresses known not to be on the path, we found that 5.4% of overstamps are actually incremented from the initial stamp, a false positive under our strawman test. And, the test would lead to false negatives for adjacent nearby routers that stamp the same time. By probing for timestamps to adjacent hops seen in a day's iPlane traceroutes, we found such cases in 6.3% of cases. We employ a different test to eliminate both these false positives and false negatives. If the second timestamp is equal to or one more than the first, we send a followup probe to D, asking for timestamps D, X, where X is the address of a machine in our lab (therefore not on the path). If D does not overstamp for X, then we declare R as being on the reverse path.

Consecutive Probes without Timestamp	Chance of Timestamp with More Probes
1	10.1%
2	6.3%
3	4.3%
4	3.1%
5	2.4%
10	0.9%

Table 3: For routers observed on traceroutes from PlanetLab to iPlane destinations, the chance that the router will timestamp a ping sent to the destination as a function of the number of previous pings it has not timestamped.

9.4 Using timestamp to establish a router is not on the path

Normally we use timestamp to establish that a router is on a path, and, if a router actually on the path fails to timestamp, we miss inferring a reverse hop. In the technique described in the last paragraphs, we use timestamp to establish that a router is not on the path from and to V, with a router failing to timestamp meaning we make a false inference that it *is* on the path to S. To test whether we can claim that R is not on V's path if it does not timestamp, for each iPlane destination D, each PlanetLab source V, and each router R on V's traceroute to D, we send a dozen timestamp pings from V to D asking for R's timestamp. Table 3 gives the likelihood of receiving a timestamp in the last $(12 - n)$ probes after not receiving one in the first n. R might not timestamp the initial probes because they were lost, because it was rate-limiting its stamps, or because the probes were load-balanced onto a path that did not include R. In practice, we send 3 probes to test whether R is on V's path. Since, when we reach this test, we already know R is on S's forward and/or reverse path to D and is adjacent to D in the network topology, a false positive will mean that we assumed V was the first hop on the reverse path, when in fact it was the last hop on the forward path. In the case when we are unable to determine a hop using timestamp, we anyway assume that last hop on the forward path is symmetric, so false positives with timestamp are equivalent to our fallback assumption.

9.5 Cases of Unreachability

Frequently, a user turns to traceroute when unable to reach a destination D, to learn how much of the path works and where the failure might be. Traceroute returns some hops, then returns nothing after the point of the failure. It generally has similar output regardless of whether the failure is on the forward path, reverse path, or both.

With our previous system Hubble, we found that, when some source *S* was unable to reach *D*, we could frequently find some other vantage point *V* that could reach *D* and that, in most of these cases, spoofed pings isolated the failure to either the forward or the reverse path [19]. However, traceroute requires the reverse path to work to receive responses from routers on the working forward path, and vice versa, so Hubble could not measure which routers were on the working path past the point when traceroute failed. The techniques we present in this paper add to that diagnostic information.

Suppose the problem is on the reverse path. Because *D* does not have a working path to *S*, *S* cannot receive *D*'s responses to RR or TS queries, so has no way to determine the initial hops back from *D*, even with the help of a spoofing node *V*. However, *S* can issue a standard traceroute, but spoof as *V*, such that the TTL-expired responses from all hops will go to *V*, yielding a complete forward path from *S* to *D*. *S* can then ping all the forward hops, to see which have paths back to it, and issue reverse traceroutes for all that do, determining a set of working paths and helping narrow down where the reverse path might be failing. Furthermore, if *S* has a premeasured reverse path from *D* that predates the failure–say, by periodically issuing reverse traceroute to a set of important destinations–then it can determine the farthest hop *h* along that path that can still reach it, as well as the first hop *h'* past *h* that cannot. At this point, it knows that either the link *h'*-*h* is down, or *h'* has changed paths.

Suppose instead the problem is on the forward path. Depending on the particular pattern of partial reachability, we may have spoofing vantage points able to reach *D*, in which case they can send spoofed RR and TS queries as *S*, and we can likely still determine the reverse path back from *D*. *S* can then check which of the reverse path hops it can reach, as well as the degree to which they diverge from the partial traceroute it can issue to *D*. We have implemented the fully spoofed forward traceroute and reverse traceroute described in this section, and we will conduct a study of their diagnostic power in future work. Currently, in cases of partial reachability, a lack of tools and vantage points often forces network operators to issue traceroutes by hand from their own networks and post the results to mailing lists [39, 27]; our techniques can provide extra information to aid in diagnosis.