Internet Scale Reverse Traceroute

Kevin Vermeulen, Ege Gurmericliler, Ítalo Cunha, Dave Choffnes, Ethan Katz-Bassett

Internet Scale Reverse Traceroute

Kevin Vermeulen
LAAS-CNRS

Ege Gurmericliler
Columbia University

Italo Cunha
Universidade Federal de Minas Gerais

David Choffnes
Northeastern University

Ethan Katz-Bassett
Columbia University

ABSTRACT

Knowledge of Internet paths allows operators and researchers to better understand the Internet and troubleshoot problems. Paths are often asymmetric, so measuring just the forward path only gives partial visibility. Despite the existence of Reverse Traceroute, a technique that captures reverse paths (the sequence of routers traversed by traffic from an arbitrary, uncontrolled destination to a given source), this technique did not fulfill the needs of operators and the research community, as it had limited coverage, low throughput, and inconsistent accuracy. In this paper we design, implement and evaluate REVTR 2.0, an Internet-scale Reverse Traceroute system that combines novel measurement approaches and studies with a large-scale deployment to improve throughput, accuracy, and coverage, enabling the first exploration of reverse paths at Internet scale. REVTR 2.0 can run 15M reverse traceroutes in one day. This scale allows us to open the system to external sources and users, and supports tasks such as traffic engineering and troubleshooting.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Network monitoring**.

KEYWORDS

Internet Measurements; Reverse Traceroute; Internet scale

ACM Reference Format:
Kevin Vermeulen, Ege Gurmericliler, Italo Cunha, David Choffnes, and Ethan Katz-Bassett. 2022. Internet Scale Reverse Traceroute. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC '22), October 25–27, 2022, Nice, France.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3517745.3561422

1 INTRODUCTION

Most Internet paths are asymmetric [30], and so operators troubleshooting problems and researchers seeking to understand Internet routing want visibility into both directions to obtain a complete picture. However, traditional tools such as traceroute only measure the forward path from the user, leaving the reverse path unknown [73]. The invisibility of reverse paths complicates troubleshooting [73], hinders industry efforts to improve client performance [56] and academic efforts to localize Internet outages [51], and forces systems to rely on unrealistic assumptions of path symmetry [83].

To overcome this longstanding impediment, researchers and operators need a way to measure reverse paths with *high accuracy*, *high coverage*, *high throughput*, and *low latency*, similar to what is possible with traceroute. We need: (1) the ability for users to measure paths to their own hosts from arbitrary remote hosts, without access to those remote hosts to run commands; (2) large-scale measurements of reverse paths from hosts across the Internet towards distributed vantage points in a reasonable time, *e.g.,* 20 million measurements per day like Ark does for forward traceroute [1]; and (3) the ability to issue on-demand measurements to support operations and dynamic experiments, as one could with forward traceroutes from RIPE Atlas, the cloud, and one's own hosts.

Although earlier research developed Reverse Traceroute [52] (named REVTR 1.0 hereafter), a technique to measure reverse paths from uncontrolled destinations, it did not satisfy these goals. It required issuing a large number of probes, which limited throughput. The 2010 paper only measured a few thousand reverse paths, and even our reimplementation of the system, which adds better vantage points, optimized code, and caching/reuse of intermediate measurements, can measure only a few hundred thousand reverse paths per day (§5.2). Further, although the overall accuracy was high, REVTR 1.0 sometimes returned incorrect paths and did not have a way to know whether a particular measurement was trustworthy, making it hard to trust its results and rendering the system unsuitable for operational use (according to our contacts at a hypergiant that deployed a system based on that earlier research paper). Finally, the approach required complex dynamic orchestration of its distributed vantage points and only allowed measurements back to PlanetLab and M-Lab vantage points, limiting its usefulness in practice since measuring other routes would require deploying a similarly complex system.

We make the following contributions:

REVTR 2.0, a system that can measure reverse paths at Internet scale. While the basic measurement technique (called Reverse Traceroute) is the same as REVTR 1.0, we identify explicit tradeoffs among coverage, accuracy, and throughput and select tradeoffs to meet our goals. We build a system using new measurement techniques, measurement studies, and system optimizations. Because it is a complex distributed system requiring coordination of vantage points around the world, we operate REVTR 2.0 as a service: researchers and operators can use our deployment to measure reverse paths back to their own sources (Appx. A).

Throughput: New techniques that enable REVTR 2.0 to sustain a throughput of 173 reverse traceroutes per second, 43× higher than REVTR 1.0 and similar to Ark's [1] rate of forward traceroutes. Our new techniques significantly reduce the number of probes issued to measure a reverse path. The most efficient ways Reverse Traceroute discovers reverse hops are by issuing record route

IMC '22, October 25–27, 2022, Nice, France

ACM ISBN 978-1-4503-9259-4/22/10…$15.00
https://doi.org/10.1145/3517745.3561422

probes, which measure the first nine hops (including the forward path), making fast discovery of nearby vantage points key; and by intersecting known routes in a *traceroute atlas*, short circuiting the need to issue more probes.

Our first technique improves record route vantage point selection by grouping vantage points by their ingresses into a destination prefix, avoiding wasteful redundant measurements. Our second technique increases the set of potential intersections with the traceroute atlas by revealing additional IP-address aliases that are likely to be observed in reverse traceroutes. Finally, we perform a measurement study to find the right balance between atlas size (more traceroutes provide more opportunities for intersections), atlas freshness (stale measurements introduce opportunities for incorrect inferences), and atlas coverage (smart traceroute selection may yield more intersections), given a limited probing budget.

Accuracy: New techniques that enable 92.3% of REVTR 2.0's reverse traceroutes to exactly match the AS path of a direct traceroute issued from the destination and an additional 6.1% to match except for unresponsive hops ("*"), compared to 81.8% for REVTR 1.0,. Router-level accuracy is hard to assess because it is hard to infer which IP addresses belong to the same router and because load balancing causes multiple router paths, but our analysis suggests that REVTR 2.0 does not introduce errors.

We improved accuracy by discarding reverse traceroutes likely to contain errors and by flagging possible missing/unresponsive hops. When REVTR 1.0 is unable to measure a reverse hop, it issues a forward traceroute to the current hop, assumes the last link is symmetric, and continues the reverse traceroute from the penultimate hop. We conduct a measurement study and find that assuming symmetry on an intradomain hop is generally correct, while assuming symmetry on an interdomain hop is not. We therefore allow Reverse Traceroute to assume symmetry only on intradomain hops. By discarding measurements rather than risking an incorrect assumption of interdomain symmetry, REVTR 2.0 achieves high accuracy so that its measurements can be used with confidence.

Coverage: REVTR 2.0 can measure reverse paths from destinations in 39,544 ASes that host 92.6% of Internet users [7], despite discarding measurements that would require assuming interdomain symmetry. This coverage dwarfs RIPE Atlas (3,682 ASes).

Demonstration of how REVTR 2.0 enables outside sources to measure routes and use them to perform traffic engineering: REVTR 2.0 can realize our goal of on-demand measurements in support of operations and can incorporate external sources and users. We demonstrate this via a traffic-engineering case study with the PEERING testbed [67] in which REVTR 2.0 measured reverse routes to inform announcement reconfigurations, which in turn led to better load balancing and reduced latency.

A large-scale study of Internet path asymmetry: To demonstrate how REVTR 2.0 supports our goal of large-scale topology mapping, we conduct the most comprehensive study on Internet path asymmetry to date with 30M pairs of forward and reverse path measurements, significantly expanding on prior work that considered 120K paths between RIPE Atlas vantage points [30]. We find that only 53% of paths are symmetric even at the coarse AS-level granularity, further demonstrating the need for an Internet-scale Reverse Traceroute system.

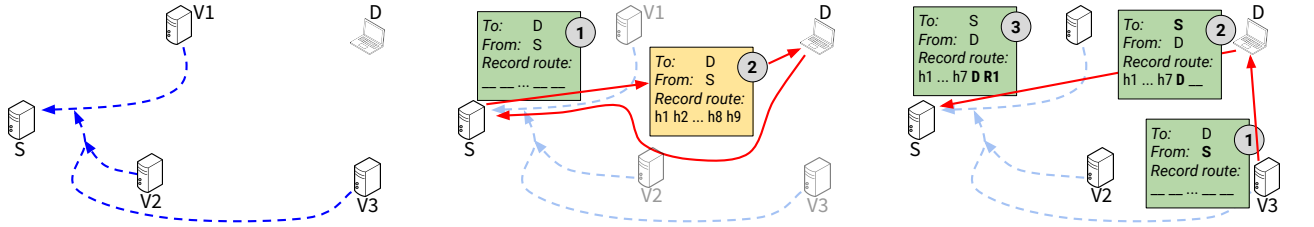
2 BACKGROUND

This section reviews our 2010 Reverse Traceroute technique. That earlier work has further details [52]. Whereas (forward) traceroute measures the path from a source to an uncontrolled destination, the goal of Reverse Traceroute is to measure the path from an uncontrolled destination back to the source (from D to S in Figure 1). To build the reverse path from D to S, Reverse Traceroute relies on the assumption that most Internet routing is destination-based (which we validated in [34] and in Appx. E), such that the next hop from a given router depends only on the destination (which here is S). This allows Reverse Traceroute to piece together the path hop-by-hop to S: when it discovers a new reverse hop, that *current* hop is set as the new destination D', and Reverse Traceroute measures the reverse path from D' until the path reaches S. To find the next reverse hop, Reverse Traceroute tries different measurement techniques that rely on a set of distributed vantage points. Figure 1, based on Figure 3 of our 2010 paper, recaps these different techniques in the order Reverse Traceroute tries them:

Intersecting a traceroute (Fig. 1a): If a traceroute to the source *intersects* the current hop (includes that hop or an IP alias of it), destination-based-routing means that we can assume that the rest of the reverse path follows the traceroute from the intersection to S. To create a set of potential intersections, Reverse Traceroute maintains an atlas of traceroutes from a set of vantage points to the source. In Figure 1a, the traceroute atlas includes traceroutes from vantage points V1, V2 and V3, none of which intersect the current hop (D, at this point).

Record route (RR) (Figs. 1b to 1d): If no intersecting traceroute can be found from the current hop, the source S sends the current hop an ICMP echo request with the IP record route option enabled. This option instructs routers on the path to record their IP addresses in the packet header, which has space for up to nine addresses. When the current hop replies with an ICMP echo reply, it copies the IP options into the response. If fewer than 9 hops were recorded on the forward path, the remaining slots can be filled with reverse hops. If S is more than 8 hops away from the current hop (Fig. 1b), Reverse Traceroute tries to find a vantage point within record route range of the current hop, using that vantage point to send the packet to the current hop while spoofing as S, causing the current hop to reply to S along the desired reverse path. In Figure 1c, V3 is 8 RR hops away from D, so when D responds to S, Reverse Traceroute uncovers the reverse hop R1 towards S. Then, in Figure 1d, V2 is 7 RR hops away from R1, uncovering the reverse hops R2 and R3.

Timestamp (TS) (Figure 1e): If Reverse Traceroute is unable to measure a reverse hop using record route, it tests *adjacencies* of the current hop in traceroute topologies as possible next reverse hops. For each adjacency, Reverse Traceroute sends an IP timestamp tsprespec packet to the current hop. The IP timestamp tsprespec option allows the sender to specify up to four IP addresses, and each IP address will record its timestamp only if previous addresses already recorded their timestamp. Reverse Traceroute specifies the IP addresses ⟨current hop, adjacency⟩. If the response includes a timestamp for both hops, then the adjacency must have recorded its timestamp after the current hop, implying that the adjacency is on the reverse path. On Figure 1e, after having discovered R1, R2 and R3 with record route, Reverse Traceroute sends an IP timestamp



(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 public traceroutes 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 1: How 2010 Reverse Traceroute technique measures a reverse path using various techniques and vantage points [52].

tsprespec packet to R3 with the prespecified addresses ⟨R3, R4⟩. As R4 put its timestamp after R3 has been reached, R4 is actually on the reverse path to S. At this point, the reverse path Reverse Traceroute is measuring from D intersects the traceroute from V1 at R4 in Figure 1f, and Reverse Traceroute assumes the reverse path from D to S follows V1's traceroute from R4 to S.

Assuming symmetry: If no techniques uncover the next reverse hop, Reverse Traceroute issues a traceroute from source to destination and assumes the penultimate hop is the next reverse hop.

3 GOALS

Our goals are informed by our experiences designing, operating, and/or evaluating multiple Reverse Traceroute implementations over the last decade, including REVTR 1.0 for the 2010 paper [52] and a deployment at a large cloud provider. These implementations had limitations with respect to needs of researchers and operators.

Coverage: A Reverse Traceroute system must be able to measure paths of interest. Broad coverage requires widespread router support for IP options, and it requires vantage points near (in terms of hops) routers on reverse paths. It also benefits from a rich traceroute atlas to intersect reverse paths early. The system must allow external users to add their sources to measure paths towards them.

The 2010 paper did not evaluate Internet-wide coverage [52], but REVTR 1.0 had a major limitation: it could only measure paths for its (PlanetLab/M-Lab) vantage points. As such, it was not useful to operators who care about paths traversing their networks. Adding outside vantage points requires solving research challenges that the 2010 paper did not address, since using a vantage point requires knowledge of routing to and from it: How can this knowledge be quickly bootstrapped? Given feasible probing rates, how should the system balance using more measurements to bootstrap more effectively, versus saving probing budget to service reverse traceroute requests or add more vantage points?

Accuracy: A system must return trustworthy results. Since REVTR 1.0 could not support outside sources, a cloud provider consulted with us to deploy their own instantiation. Reverse Traceroute sometimes made errors in path measurements [52] (e.g., due to assuming symmetry), and so the results could not be relied on for operational decisions, resulting in the instantiation being decommissioned.

Throughput: REVTR 1.0 was designed to support proof-of-concept experiments but not high measurement throughput. To serve researchers and operations, REVTR 2.0 should support two common use cases of traceroute: large-scale *topology mapping* and *on-demand measurements*.

Topology mapping. REVTR 2.0 should provide reverse path measurements analogous to what Ark provides for forward paths. Ark measures from ≈110 vantage points to all ≈800,000 IPv4 BGP prefixes at a rate of ≈20M traceroutes per day [1]. Recent work found that >90% of paths are still valid after 10 days [38], and so our goal is to support remeasuring paths every 10 days. Measuring from 800,000 destinations to the 146 M-Lab sites in 10 days requires a throughput of ≈11.7M reverse traceroutes per day.

On-demand measurements. Exact needs vary, but two attributes suffice to support a large range of use cases. REVTR 2.0 should support reverse path measurements to users' own sources, rather than being limited to measure paths to M-Lab vantage points. Second, users should be able to characterize Internet routing to their sources in a short period of time.

Akamai optimizes anycast routing worldwide by pinging 15,300 routers at key convergence points on routes to clients [82], then

adjusting routes to see how the pings change. Because this scale of destinations suffices to capture the routing of a major Internet content provider, it may also suffice for other use cases such as identifying hijacks [83], locating failures [53] and causes of route changes [49], and identifying sources of spoofed DDoS attacks [36].

Since these use cases either rely on route engineering or on understanding route changes, REVTR 2.0 should be able to measure these routes in roughly the time period that traffic engineering techniques are given to converge. Announcements every 15 minutes were subject to route flap damping [41], meaning that faster changes impede traffic engineering, and so we want to support quickly bootstrapping outside sources and issuing ≈15,000 reverse traceroutes within 15 minutes. This rate equates to ≈1.4M per day, but, in practice, most studies are more conservative (waiting 90 [82] or 120 [19, 25, 53, 66] minutes between announcements), and so 1.4M per day will suffice to support multiple parallel uses.

Combining the two use cases, our goal is a system that can measure to users' sources and that supports at least 11.7M reverse traceroutes per day for topology mapping and 1.4M per day for on-demand measurements (13.1M per day total). If the system becomes popular, we (or others) can add parallel deployments. In contrast, REVTR 1.0 only supported PlanetLab and M-Lab sources [52] and could only measure a few hundred thousand paths per day (§5.2).

4 BUILDING AN INTERNET SCALE REVERSE TRACEROUTE SYSTEM

With these goals in mind, we decouple the basic Reverse Traceroute technique (§2) from a set of key design questions that have to be answered to arrive at a concrete instantiation of a Reverse Traceroute system. For each question, Section 4.1 explains the tradeoffs and insights that enable a solution that meets our goals and contrasts our design with the 2010 Reverse Traceroute implementation.

We strive for a system with the coverage and accuracy to be useful. However, we consider tradeoffs such as the limitation on throughput imposed by employing many additional measurements to achieve marginal gains in coverage/accuracy. It would also be better to slightly reduce coverage by not returning low-confidence measurements, if the result is a system that achieves high accuracy for the paths it does measure. Since system throughput has an inverse relationship to the packets sent per reverse traceroute measurement, we strive to reduce probes per measurement by moving online measurements offline, by caching measurements for reuse, by minimizing the number of probes required to find one that uncovers reverse path information, and by maximizing the reverse hops uncovered per probe.

Our main contribution in this paper is an Internet-scale Reverse Traceroute system that is available as a service to the community (Appx. A). We have made progress on addressing the limitations of the system described in our 2010 Reverse Traceroute paper, both in our earlier work [34, 38, 39] and in this paper. This paper is the culmination of that effort, building on and extending our earlier work to arrive at the first system capable of measuring reverse paths at Internet scale. Table 1 lists the key insights across this body of work and how they benefit REVTR 2.0. Insights 1.1 to 1.3 are the key enablers of Reverse Traceroute (in brief):

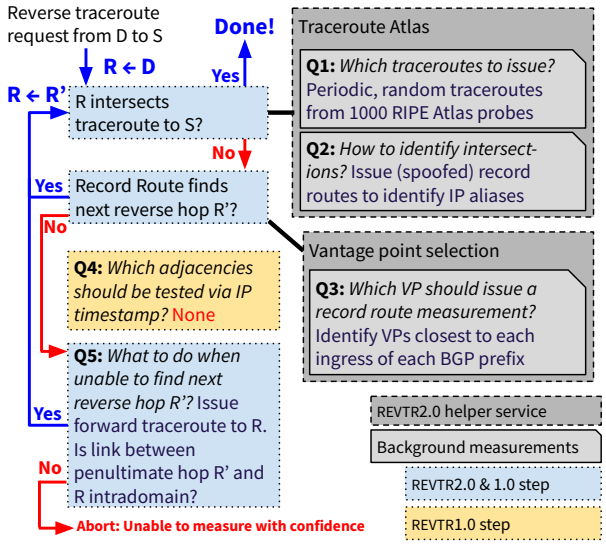


Figure 2: Overview of the REVTR 2.0 control flow.

Insight 1.1: Because most routing is destination-based, the reverse path can be measured hop-by-hop.
Insight 1.2: IP options can measure reverse hops.
Insight 1.3: Spoofing enables using the best-located vantage point.

Our 2010 paper did not assess how often these claims held. This paper evaluates them, finding that 6.6% of hops violate destination-based routing (ignoring load balancing, which does not impact accuracy), but only 1.1% hurt REVTR 2.0's AS-level accuracy (Appx. E); 78% of ping-responsive destinations respond to options (Appx. F); and, without spoofing, reverse hops can be measured for 32% of ⟨`source, destination`⟩ pairs (of destinations that respond to options), whereas spoofing enables measurements for 63% (Appx. F).

Section 4.1 highlights our other insights and how they inform how REVTR 2.0 answers the design questions to arrive at a system that meets our goals. Figure 2 shows an overview of the REVTR 2.0 system, including steps to measure hops, helper services to support those steps, and the corresponding design questions (**Q#**). The end of Section 4.1 walks through the figure end-to-end.

4.1 Design questions

Q1. Which traceroutes to issue for the traceroute atlas? *REVTR 2.0 measures from 1000 random RIPE Atlas VPs to each source daily, replacing redundant traceroutes with new VPs each day.*

Insight 1.4: Paths are stable enough that the atlas need not be refreshed frequently and can be tuned across days [38].

Insight 1.5: Routes generally form a tree rooted at the source. It takes many traceroutes to cover rarely-used branches far from the source, but much of the benefit comes once one has issued sufficient measurements to cover widely-shared branches closer to the source, which random measurements can achieve.

Tradeoffs: RIPE Atlas provides path diversity but has severe rate limits [29]. By allocating its probing budget to a relatively small atlas per source with daily refreshing to limit staleness, REVTR 2.0 balances accuracy, coverage and throughput. Only 0.7% of reverse

Table 1: The Internet-scale Reverse Traceroute system in this paper relies on and is the culmination of insights, measurement techniques, and measurement studies in this paper and our earlier work with collaborators [34, 38, 39, 51, 52, 60, 68].

Insight	*Impact on* REVTR *2.0*
Basic Reverse Traceroute building block techniques	
Insight 1.1: Since most routing is destination-based ([34], Appx. E), the reverse path can be measured hop-by-hop ([52]).	*Accuracy*: Ignoring load balancing, which does not impact accuracy, 6.6% of hops are not destination-based. Only 1.1% hurt AS-level accuracy (Appx. E).
Insight 1.2: IP options can measure reverse hops ([39, 52, 60, 68]).	*Coverage*: 78% of ping-responsive destinations respond to options (Appx. F).
Insight 1.3: Spoofing decouples the forward and reverse paths to allow use of vantage points in the best positions, with the request sent from a different vantage point than where the response is received ([51, 52]).	*Coverage*: Without spoofing, reverse hops can be measured for 32% of ⟨source, destination⟩ pairs (of destinations that respond to record route), whereas with spoofing reverse hops can be measured for 63% (Appx. F).
Balancing coverage and freshness of the traceroute atlas, given a limited measurement budget	
Insight 1.4: Most paths are stable, so path measurements can be cached for extended periods ([38]).	*Throughput and coverage*: By caching traceroutes for a day rather than refreshing more frequently, REVTR 2.0 covers more paths with a fixed budget, intersecting traceroutes sooner (enabling it to issue 26% as many packets as REVTR 1.0, §5.2.4). *Accuracy*: Only 0.7% of reverse traceroutes use a stale traceroute (Appx. D.2.2).
Insight 1.5: A relatively small number of randomly selected vantage points suffice to create a traceroute atlas by uncovering widely used routes (Appx. D.2.1). Rarely used routes require many measurements to uncover but, being rarely used, do not yield much additional value.	*Coverage*: Traceroutes from 1000 random vantage points, refreshed daily after replacing those that proved not useful, provide 93% of the value of the optimal 5000 (Appx. D.2.1). *Throughput*: This atlas provides 56% of the hops on a REVTR 2.0 measurement, reducing the online measurements needed (Appx. D.2.1).
Refining techniques for efficient IP options probing, yielding higher throughput	
Insight 1.6: Aligning record route and traceroute measurements to determine whether/where they intersect is hard. Instead, REVTR 2.0 determines *a priori* which record route hops it will see each hop in a set of traceroutes (§4.2), obtaining better tradeoffs for our use case.	*Throughput*: Our technique finds more intersections sooner, saving 5.5% of the probing overhead (§5.2.4). Its measurements are offline and scale only with the size of the traceroute atlas, without requiring additional measurements as throughput increases.
Insight 1.7: The flattening of the Internet and the expansion of M-Lab put more destinations within record route range ([39], Appx. F).	*Coverage*: 63% of destinations are in range, vs. 12% in 2011 (Appx. F). *Throughput*: As vantage points are closer (*e.g.,* 39% of destinations are within 4 hops, vs. 16% in 2016), each reverse traceroute requires fewer probes, increasing throughput.
Insight 1.8: To find the closest vantage point to a destination, it suffices to probe the destination once per ingress into the destination's prefix, from the vantage point closest to that ingress (§4.3).	*Throughput*: Whereas REVTR 1.0 used 20% of probing budget offline to find vantage points near destinations, REVTR 2.0 reduces the overhead to 3%, using saved probes to increase Reverse Traceroute throughput. It also uses many fewer online probes via smarter VP selection (9 RR probes per path (mean) vs. to 29 for REVTR 1.0 (§§ 5.2.4 and 5.3)), which translates into lower latency and higher throughput.
Insight 1.9: Although the IP timestamp option can measure reverse hops ([52, 68]), each hop typically requires multiple probes. The probing overhead is not worth the limited additional coverage.	*Throughput*: By not using timestamp, REVTR 2.0 requires ≈34% fewer online probes (§5.2.4), increasing throughput. *Coverage:* By not using timestamp, REVTR 2.0 loses less than 1% of coverage (§5.2.3).
Trading off coverage for high accuracy	
Insight 1.10: For operational use, it can be better to only provide answers when they are trustworthy than to maximize coverage but have an unknown subset of answers be incorrect.	*Coverage*: REVTR 2.0 only returns results for 78% of attempted paths. *Accuracy*: Returning the additional paths would require assumptions that are only correct for 57% of cases (§4.4).

traceroutes intersect a stale traceroute (Appx. D.2.2). and the atlas converges in 5 iterations to a near-optimal set of traceroutes (Appx. D.2.1). This near-optimal coverage improves throughput, as Reverse Traceroute intersects traceroutes earlier, requiring fewer record route probes.

REVTR 1.0: REVTR 1.0 used the 200 PlanetLab sites that existed, which were biased toward educational networks, and 1200 web-based traceroute servers hosted in 186 networks, which are severely rate limited. All vantage points issued traceroutes to the sources when a reverse traceroute was requested, which incurred significant overhead and limited throughput.

Q2. How should intersections between record route measurements and the traceroute atlas be identified?

Routers may return different IP addresses to RR and traceroute measurements, complicating detecting intersections.

REVTR 2.0 issues RR probes to each hop in the traceroute atlas to identify reverse RR hops used by that hop (§4.2). A subsequent RR measurement intersects the traceroute if it includes any of the reverse RR hops.

Insight 1.6: While router IP alias information can be used to determine if paths intersect, approaches have high overhead and poor coverage [17, 54], and aligning RR and traceroute is hard [70]. We sidestep these challenges by focusing on identifying just the IP addresses used on traceroutes and RR probes toward the source. By designing a solution to our specific problem of determining intersection, we obtain better tradeoffs for our use case.

Tradeoffs: REVTR 2.0's approach promotes high throughput by moving all measurements offline, which is impossible with earlier techniques that require knowing both IP addresses to check intersections, and by identifying more intersections, reducing record route measurements per path. The number of measurements needed by

this technique scales with the size of the traceroute atlas, without requiring additional measurements as throughput increases. Finding more intersections promotes accuracy and coverage as well.

REVTR 1.0 combined alias resolution datasets [22, 55, 59, 70]. REVTR 1.0 could not identify intersections for IP addresses not in these datasets and uncovered for the first time during a reverse traceroute. Missing intersections increases the number of probes needed to measure a reverse path, limiting throughput and introducing opportunities for inaccuracies.

Q3. Which vantage point(s) should issue a particular spoofed record route measurement?

The closer a vantage point is to a destination, the more hops it can reveal in one probe, reducing the number of probes to measure a reverse path. In addition to the order, one can change batch size or the stop condition to balance between using more vantage points to maximize hops revealed (potentially reducing the need for more iterations) or fewer to minimize overhead in the current iteration.

REVTR 2.0 identifies the closest M-Lab vantage point to each ingress into each network, then uses those closest vantage points to probe any destination in the network (§4.3).

Insight 1.7: Colocation facilities are close to many networks in today's flattening Internet, providing the potential to reveal more reverse hops per RR measurement, and they tend not to filter the spoofed packets Reverse Traceroute requires. M-Lab provides access to VPs in such facilities around the world.

Insight 1.8: Each prefix has a fixed set of ingress routers. Because of destination-based routing, a vantage point will generally use the same ingress to reach all destinations in a prefix, and the path from an ingress to a destination will generally be the same regardless of source. Therefore, to maximize the number of reverse hops obtained from an RR measurement, it suffices to probe the destination once per ingress, from the M-Lab vantage point closest to that ingress.

Tradeoffs: Even when PlanetLab was active, it provided negligible incremental benefit over M-Lab's record route coverage [39]. Whereas REVTR 1.0's approach requires dedicating 20% of the probing budget to offline measurements that support answering this question, REVTR 2.0's approach reduces the overhead to 3%, saving more probing budget to increase Reverse Traceroute throughput. REVTR 2.0 finds a vantage point in range for 99% of destinations that have one, trading off this slight loss in coverage for much lower online probing overhead via smarter VP selection (an average of 9 RR probes per path, compared to 29 for REVTR 1.0 (§§ 5.2.4 and 5.3)), which translates into lower latency and higher throughput.

REVTR 1.0 relied on costly background measurements to order vantage points, and it would try them all until one reached the destination, possibly wasting budget when the vantage points are not within range. Finally, PlanetLab nodes were mainly at universities, and hence far from the core, resulting in limited coverage given record route's nine hop limit [39].

Q4. Which adjacencies should be tested via IP timestamp option measurements? *REVTR 2.0 does not use timestamp.*

Insight 1.9: Most routers that support IP timestamp also support record route. The flattening Internet and expansion of M-Lab place a vantage point within RR range of most routers [39].

Tradeoffs: Timestamp provides little benefit in accuracy and coverage (<1%), so is not worth the overhead and attendant reduction in throughput (an average of 5 timestamp probes per path for REVTR 1.0, §§ 5.2.3 and 5.2.4, and Appx. D.1).

REVTR 1.0 used every adjacency in the iPlane dataset [2].

Q5. What should Reverse Traceroute do when it is unable to measure a hop? *REVTR 2.0 issues a forward traceroute to the last measured hop. If the last link is intradomain, REVTR 2.0 assumes it is traversed symmetrically on the reverse path. If it is interdomain, REVTR 2.0 gives up on measuring the path.*

Insight 1.10: For operational use, it can be better to only return paths when they are trustworthy than to maximize coverage but have an unknown subset of untrustworthy paths.

Tradeoffs: REVTR 2.0 is able to measure reverse paths for only 78% of paths (compared to 100% for REVTR 1.0), as measuring them would require assuming symmetry of interdomain links (§5.2.3). However, such assumptions are only correct for 57% of the cases, and so REVTR 2.0 trades off coverage to achieve high accuracy. Conversely, assuming symmetry of an intradomain link is correct in 90% of the cases, making these symmetry assumptions safe (§4.4).

REVTR 1.0 issues a forward traceroute to the last hop on the reverse path and always assumes symmetry. Assuming symmetry sometimes returns incorrect paths, and so every measurement had to be treated with suspicion.

Summary and walk-through of REVTR 2.0. Figure 2 depicts how our answers to the design questions fit together to measure a reverse traceroute from a destination D to a source S. We walk through the figure with pointers to the insights from Table 1 that inform each step. Table 1 highlights the quantitative impact on REVTR 2.0 of each insight. The system pieces together the path incrementally from D back to S (Insight 1.1). At any time, it is trying to discover hops from the current hop R (initialized to D) towards S. REVTR 2.0 first checks whether R is on a known route to S as measured daily in the background for the traceroute atlas (Q1, Insights 1.4 and 1.5). To increase the chance of finding an intersection, the system also issues background record route measurements to the hops in the atlas traceroutes to learn which IP addresses to expect if aliases of the hops are later encountered as part of a reverse traceroute (Q2, Insight 1.6). If R intersects a traceroute to S, REVTR 2.0 uses the rest of the traceroute to complete the reverse traceroute (Insight 1.1). If not, REVTR 2.0 tries to uncover a reverse hop(s) from R with record route (Insight 1.2), spoofing as S (Insight 1.3) from the M-Lab vantage points closest to the ingresses into the prefix of R, which it determines using background measurements (Q3, Insight 1.8). If record route uncovers a reverse hop R', REVTR 2.0 sets the new current hop as R' and returns to the first step. If record route does not uncover any hops, REVTR 2.0 does not issue timestamp measurements, unlike REVTR 1.0 (Q4, Insight 1.9), and instead performs a forward traceroute to R and considers the penultimate hop R'. If the link (R',R) is intradomain, REVTR 2.0 sets the new current hop as R' and returns to the first step (Insight 1.1). If interdomain, REVTR 2.0 aborts the reverse traceroute as it cannot confidently measure the path (Q5, Insight 1.10).

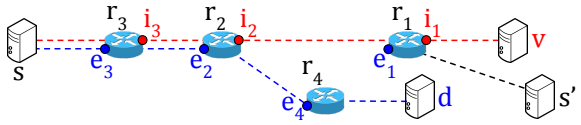


Figure 3: Traceroutes typically reveal ingress interfaces (red) while RR reveals other addresses (blue). We send RR probes from S (or spoofed as S) to traceroute interfaces (i_1, i_2, i_3) to uncover the interfaces (e_1, e_2, e_3) that would be seen by a later reverse traceroute (*e.g.,* from D to S) that uses RR.

4.2 Intersecting record route and the traceroute atlas (Q2)

We develop a technique to identify a priori what IP addresses to expect in RR probes if a later reverse traceroute intersects the traceroute atlas. After performing an atlas traceroute, we send RR probes to each traceroute hop to reveal the router's IP address revealed by RR probes towards the source. Later, if a reverse traceroute includes one of these addresses, we conclude that it intersected the atlas traceroute. On the other hand, if a reverse traceroute uncovers a *new* IP address, we infer that it does not intersect traceroutes in the atlas, avoiding the need to perform aliasing at runtime.

In Figure 3, to reveal the interfaces that may be observed by RRs during a reverse traceroute, we send RR pings to i_1, i_2, i_3 from s or spoofing as s from vantage points close to each interface. Here the spoofed RR ping from s' to i_1 reveals $[i_1, e_1, e_2]$. Later, the reverse traceroute from d to s has e_2 on the reverse path, allowing us to intersect the atlas traceroute from v to s and the reverse traceroute from d to s at r_2.

4.3 Selecting vantage points to issue spoofed record route probes (Q3)

Routes from different sources that share the same ingress to a BGP prefix converge on their way to the ingress, and routes to different destinations inside the BGP prefix diverge after traversing their ingress link. So, probing a destination from the closest vantage point to each of the destination prefix's ingresses suffices to find the closest vantage point to the destination—the vantage point capable of uncovering the most reverse hops via a spoofed record route measurement. Using additional vantage points would be redundant.

Identifying ingresses. REVTR 2.0 issues record route measurements every week from each of its vantage points to each IPv4 BGP prefix for which responsive destinations can be identified [33] in order to identify the prefix's ingresses. Identifying the ingresses of a BGP prefix is complicated by measurement artifacts that can obscure the topology, such as the fact that different routers stamp RR packets with different types of IP addresses, *i.e.,* inbound, outbound, loopback, or even private IP addresses [70]. As a result, the ingress of a BGP prefix on an RR path might not be the first IP address in that prefix. Figure 4 shows an example, using letters to represent prefixes and numbers to identify addresses. Given an RR path $[Y_1, X_1, D_4, D_1]$ from V_1 to D_1, using the first IP address in the BGP prefix would identify D_4 as the ingress, although X_1 is the real ingress, which we can not identify easily because X_1 is not in the BGP prefix. A mistake in determining the ingresses can hurt REVTR 2.0 performance. If REVTR 2.0 infers that vantage points share an ingress whereas they do not, some vantage points able to reveal

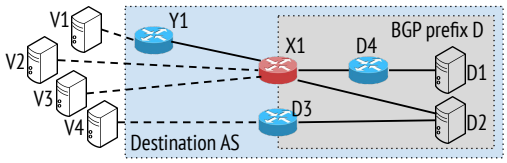


Figure 4: Example of topology where it is non-trivial to find the ingress of a BGP prefix D with a path revealed with RR, as the ingress router (at the border of D) responds with X_1, an IP address in prefix X.

reverse hops may be skipped, possibly hurting accuracy, coverage, and throughput. If it infers that vantage points do not share an ingress but they do, it can issue redundant measurements, adding probing overhead and reducing throughput.

Probing multiple destinations in a prefix to identify possible ingresses (candidates): If the only destination in D that we probe is D_1, D_4 appears to be V_1's ingress (first IP in the prefix). However, D_4 is not on the path to D_2. A better choice would be X_1, on the path to both D_1 and D_2. To guard against choosing a hop that is not traversed by all paths from a VP to destinations in that prefix (*i.e.,* that is after the real ingress), we send probes to two destinations in the BGP prefix and consider as ingress candidates IP addresses on both paths (up to and including the first IP address in the BGP prefix). The ingress candidates would be $\{Y_1, X_1\}$ for V_1, $\{X_1\}$ for V_2 and V_3, and $\{D_3\}$ for V_4. For 87.2% of prefixes, measurements to additional destinations went through the same candidates, suggesting two usually suffices.

Selecting from candidates with views from multiple VPs: REVTR 2.0 greedily selects candidates to (set) cover vantage points. In Figure 4, we would start by picking X_1 as it covers three VPs (V_1, V_2 and V_3), and then D_3, which covers the remaining VP (V_4). If multiple ingresses are tied for covering the most VPs, we choose one at random.

Coping with destinations that do not stamp RR packets: If a packet does not include addresses from the destination prefix, it could be because it was not reached, or because routers in the destination prefix do not stamp RR packets (at least not with addresses from the destination prefix). We developed heuristics that still identify ingresses in some of these cases. We infer that a RR has reached the destination if it contains a loop, which happens when a hop h is traversed on the forward path, the RR reaches the destination, and hop h is traversed again on the reverse path (Appx. C). With this heuristic, REVTR 2.0 is able to find ingresses for 97.7% of the BGP prefixes with at least one vantage point in range.

Ordering and batching measurements: When measuring reverse paths, REVTR 2.0 issues measurements only from the vantage point closest to each ingress, ordering the ingresses by the number of vantage points that use them. For the 2.3% of prefixes without identified ingresses, REVTR 2.0 ranks vantage points within 8 hops by their mean distance to the two destinations. REVTR 2.0 issues measurements from a batch of 3 vantage points at a time (§5.3), stopping when a batch reveals one or more reverse hops, or all ingresses have been tested. If the measurement from a vantage point does not go through the expected ingress, REVTR 2.0 tries the next closest vantage point to the ingress. If five vantage points in a row fail to uncover the ingress, it gives up on that ingress.

Given that we have shown that routes are generally destination-based and stable (Insights 1.1 and 1.4), REVTR 2.0's approach is based on a solid topological understanding of routing, will reliably identify and try the most valuable vantage points, and provides logical stopping conditions. Section 5.3 demonstrates that it works well in practice.

4.4 When unable to measure a hop, assume intradomain hops are symmetric (Q5)

As Reverse Traceroute is measuring a path back incrementally from the destination towards the source, it will be unable to measure from the current hop if the hop does not appear on any paths in the traceroute atlas and either does not respond to record route or is not found to be within range of any vantage points. At this point, Reverse Traceroute can either make a guess for the next hop or give up. REVTR 2.0 runs a traceroute from the source to the current hop and assumes that the penultimate hop of the traceroute is on the reverse path if it is in the same AS as the current hop. If it is in a different AS, REVTR 2.0 aborts the path measurement because the next hop toward the source cannot be reliably measured, as our study shows that 90% of intradomain links are traversed symmetrically, whereas it is only 57% for interdomain links.

Methodology. We evaluate how often it is correct to assume symmetry: how often is the last router on the forward traceroute from S to R also on the reverse path? To reveal reverse hops, we issue (spoofed) RR measurements to R spoofing as S, choosing the vantage points using the technique described in Section 4.3. If we uncover at least one reverse hop, we can classify the penultimate traceroute hop as on the reverse path, not on the reverse path, or unknown. The penultimate traceroute hop is on the reverse path if it or one of its aliases (Appx. B.1) appears among the reverse hops. Conversely, the penultimate traceroute hop is not on the reverse path if it is responsive to SNMPv3 (which provides reliable alias information [17]) but is not found among the reverse hops. This condition comes from the two following results: of the 30.5% of routers in the CAIDA ITDK dataset [6] that have at least one IP address that responds to SNMPv3, we find that 81.4% are responsive on all IP addresses identified by CAIDA as belonging to the router, and 94.8% respond with the same SNMPv3 identifier from all addresses, meaning that the identifier can be used to cluster aliases. Finally, if the penultimate traceroute hop does not appear in the reverse hops and is unresponsive to SNMPv3, it is unknown whether its router is on the reverse path because we lack reliable alias information.

Dataset and results. This methodology requires a dataset with many SNMPv3-responsive penultimate traceroute hops. Many router interfaces in the Internet are allocated a /30, so, if an address is responsive to SNMPv3, targeting the other address of the /30 might go through that router. Therefore, for each IPv4 address in a dataset of those that respond to SNMPv3 [17], we extract the other IP address of its /30 as a target. For each target, we apply the above methodology using 5 random M-Lab sites out of the 146, obtaining 1,530,960 paths for which we measure a reverse hop to evaluate. Table 2 shows that the penultimate hop is on the reverse path in 90% of the cases for intradomain links, whereas it drops to 57% for interdomain ones (omitting unknown cases), justifying that REVTR 2.0 only keeps hops from intradomains symmetry assumption.

	Yes	No	Unknown	$\frac{\text{Yes}}{\text{Yes+No}}$
Intradomain	0.68	0.07	0.25	0.90
Interdomain	0.42	0.32	0.26	0.57
All	0.60	0.14	0.25	0.81

Table 2: How frequently the penultimate hop of a traceroute is also on the reverse path.

IP to AS mapping. REVTR 2.0 uses a technique from our recent work [20] to perform IP to AS mapping (Appx. B.2). As REVTR 2.0 relies on mapping border routers to their ASes to decide if a link is intradomain or interdomain, we investigated using BDRMAPIT, a more complicated technique to infer the AS ownership of border routers [63]. BDRMAPIT is too slow to run online, and so we evaluated using it on REVTR 2.0's traceroute atlas (which is measured offline) to justify our decision to not use it. Since REVTR 2.0 discards reverse traceroutes rather than assuming interdomain hops are symmetric, the key question is whether using BDRMAPIT as opposed to our technique switches assumed hops (that intersect the traceroute atlas, where BDRMAPIT can be applied) from intradomain to interdomain, or from interdomain to intradomain. We found BDRMAPIT classified 0.03% of reverse traceroutes as containing an interdomain assumption of symmetry that our technique classified as intradomain. In these cases, our technique will result in REVTR 2.0 retaining suspect measurements, potentially impacting accuracy although it is also possible that BDRMAPIT is wrong or that the symmetry assumption happens to be right. BDRMAPIT also classified 0.6% of reverse traceroutes as containing intradomain assumptions of symmetry that our technique classified as interdomain. Since REVTR 2.0 would discard these measurements to avoid assuming interdomain symmetry, our simpler technique may compromise REVTR 2.0's coverage on these paths as compared to using BDRMAPIT. Please see Appendix B.2 for details. However, running BDRMAPIT on the REVTR 2.0 server on REVTR 2.0's traceroute atlas takes around 30min. During that time, the traceroute atlas is not available, leaving two possibilities: increasing staleness by using the previous atlas (reducing accuracy), or increasing the number of RR probes by not using a traceroute atlas (reducing throughput). Given these tradeoffs and the low impact of BDRMAPIT on our system, we opted not to use it.

5 EVALUATION

Overview: REVTR 2.0 improves correctness by 40% compared to performing forward traceroutes and assuming symmetry all along the forward paths as an alternative to measuring reverse paths, and improves completeness by 41% compared to using all vantage points from RIPE Atlas to measure reverse paths (§5.1). REVTR 2.0 supports a throughput of 173 reverse traceroutes per second (≈15M a day), 43× higher throughput than REVTR 1.0, and provides better accuracy for a reasonable coverage loss (§5.2). Our ingress-based technique to select vantage points requires more than 10 probes for less than 5% of BGP prefixes, compared to 28% of prefixes for REVTR 1.0 (§5.3). Our traceroute atlas provides 89% of the optimal savings with simple, randomized probing and converges to optimal in five days. Less than 1% of atlas traceroutes become stale before they are refreshed (Appx. D.2).

	Correctness	Completeness
REVTR 2.0	1.00	0.55
RIPE Atlas	1.00	0.06
Forward traceroutes + assume symmetry	0.60	0.78

Table 3: Reverse AS graph correctness and completeness if we measure reverse paths using REVTR 2.0, RIPE Atlas, or assume they are symmetric.

5.1 Large-scale evaluation

Measurements: We use REVTR 2.0 and traceroute to measure both directions of routes between a ping-responsive host in each routed BGP prefix [3] and 146 M-Lab sites. We use ISI's hitlist of ping-responsive hosts [33], but they are not necessarily record route-responsive. We run 101M reverse and forward traceroutes Dec. 8–14, 2021. We obtained complete reverse traceroutes for 31M paths, and complete forward traceroutes for 57M paths. Of the 31M complete reverse traceroutes, 7.3M (24%) contain an intradomain symmetry assumption, showing the improved coverage enabled by these (usually correct (§4.4)) assumptions. The remaining 76% contain no assumptions of symmetry, meaning that REVTR 2.0 measured them completely using a combination of record route and its traceroute atlas and demonstrating the coverage of its techniques.

Throughput: REVTR 2.0 measured ≈15M reverse and forward traceroutes per day during the collection of our large-scale campaign, or about 173 reverse traceroutes per second overall. This is a lower bound on throughput as it includes failed reverse traceroutes and the execution of a forward traceroute from the M-Lab node to each destination. We are investigating remaining bottlenecks.

Comparison with existing approaches: The map of reverse paths towards an AS can help the AS with traffic engineering [82], identifying route hijacks [83], locating routing failures [53], understanding route changes [49], and identifying sources of spoofed DDoS attacks [36]. To understand how well different approaches could enable these use cases, we compare how much of the AS-level Internet can be correctly uncovered using three techniques to obtain reverse paths: REVTR 2.0, RIPE Atlas, and issuing forward traceroutes and assuming paths are symmetric. We use our large-scale bidirectional measurements for REVTR 2.0 and forward traceroutes; for RIPE Atlas, we collect forward traceroutes to M-Lab nodes from all RIPE Atlas Probes.

For each technique, we identify the AS-level links used by each AS in the Internet to route towards a given source. RIPE Atlas traceroutes and reverse traceroutes identify correct links as they accurately measure the paths (§5.2.2), and so we compute the *correctness* of using forward traceroutes as the fraction of links inferred correctly (*i.e.,* the fraction where the assumption of symmetry is true). We also compute *completeness* as the fraction of ASes in the Internet (72,272 in a December 2021 BGP dump) for which a technique can infer an AS link used to route towards the M-Lab node (averaged across all nodes and ignoring whether the inferred link is correct).

REVTR 2.0 could measure at least one reverse path from destinations in 39,544 ASes, which host 92.6% of the Internet users. This number includes additional transit ASes seen in the reverse paths that do not host Atlas vantage points or destinations. RIPE Atlas vantage points can measure from destinations in 4,344 ASes, representing 67.1% of the Internet user base [7]. Table 3 shows that without REVTR 2.0, one can either have high correctness (1.0) but low completeness (0.06) with RIPE Atlas's limited number of vantage points, or better completeness (0.78) with low correctness (0.60) by assuming all paths are symmetric. REVTR 2.0 provides both correctness (1.0) and good completeness (0.55).

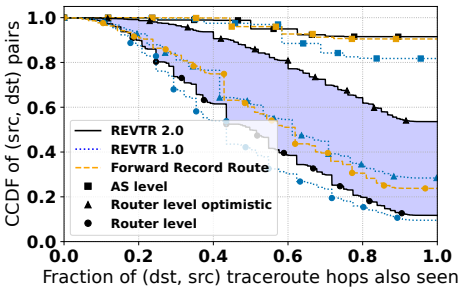
The completeness per source of REVTR 2.0 is also good: across sources, the median number of ASes from which REVTR 2.0 could measure at least one reverse path from destinations was 35.4K ASes. Of the 146 sources, 133 of them completed a reverse traceroute from destinations in more than 30K ASes. Of the 13 sources with worse coverage, 11 represent all M-Lab sites hosted in two providers, suggesting that these ASes are challenging for REVTR 2.0 to measure, perhaps for topological reasons (out of record route range) and/or configuration reasons (filtering or not responding to probes). Even these ASes are far from cloaked from REVTR 2.0: for the M-Lab source with the worst completeness, REVTR 2.0 was still able to measure routes from 19K ASes (completeness=0.26), vastly more than any existing technique with high correctness.

5.2 Comparison with REVTR 1.0

5.2.1 Setup and replicating earlier algorithms. To compare designs rather than instantiations, we: (1) use our new codebase to reimplement REVTR 1.0, using the original code as a guide; (2) let it use the same set of vantage points for spoofed RR packets as REVTR 2.0 (*i.e.,* the 146 M-Lab sites); (3) and let it use the same traceroute atlas as REVTR 2.0, but without our new IP aliasing technique (§4.2). For adjacencies for the timestamp technique (§2), we extract the links found in the Ark traceroutes from the two previous weeks before our measurements. As potential adjacencies are tested at runtime, the accuracy is not sensitive to adjacency staleness.

We compare REVTR 1.0 and REVTR 2.0 measuring 8,093 reverse traceroutes from randomly selected RIPE Atlas probes to the 146 M-Lab sites. We compare reverse traceroutes to direct traceroutes from RIPE Atlas to M-Lab to assess accuracy, despite challenges in using the direct traceroute as approximate ground truth (§5.2.2). We provide REVTR 1.0 and REVTR 2.0 with a traceroute atlas of 1000 traceroutes from randomly selected RIPE Atlas probes to each M-Lab source. When measuring a particular reverse traceroute, we do not allow the systems to access the direct traceroute or other traceroutes from the same AS. Of the 8,093 attempted reverse traceroutes, REVTR 1.0 measured 8,093 completely. Of these, 7,275 direct traceroutes reached the sources, allowing a comparison. REVTR 2.0 measures 5,682 of 7,275 (78.1%) paths completely and discards the rest to avoid assuming interdomain symmetry (§4.4).

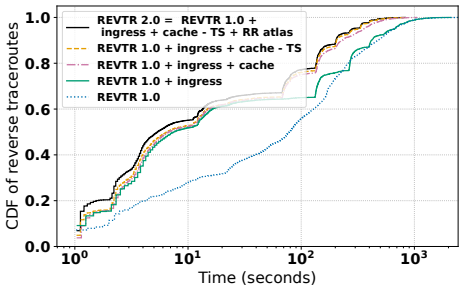
Our comparison between REVTR 1.0 and REVTR 2.0 is limited to vantage points that can run direct traceroutes to sources, and thus comprises only 8,093 reverse paths covering 1,808 ASes. This dataset and the large scale evaluation dataset (§5.1) have equivalent properties for key metrics we check including the fraction of reverse paths with symmetry assumptions, the number of hops assumed symmetric per path, and the number of RR probes sent per path, suggesting the smaller set is representative. The only figure that has a significant difference is the fraction of the reverse paths that completed successfully. It is 78% in this dataset (Fig. 5b) vs 53% in the large scale dataset (§5.1), likely because the RIPE atlas probes



(a) Accuracy

Technique	Coverage % (# paths)
REVTR 1.0	100.0% (8,093)
REVTR 2.0	78.1% (6,322)
REVTR 2.0 + TS	78.2% (6,327)
REVTR 2.0 + TS + ground truth adj.	79.2% (6,407)

(b) Coverage



(c) Latency

Figure 5: (a) REVTR 2.0 has better accuracy than REVTR 1.0. (b) It achieves this by not using untrustworthy assumptions that are likely to be inaccurate, sacrificing some coverage. (c) It also has much lower latency than REVTR 1.0.

used as destinations in this dataset are all configured to respond to record route.

5.2.2 Accuracy. Figure 5a shows the fraction of hops in the direct traceroute also seen in the reverse traceroutes at router and AS granularities. Alias resolution and IP-to-AS mapping techniques are detailed in Appendix B. There are different reasons why this fraction might be less than 1: (1) The path returned by Reverse Traceroute is wrong, because it was forced to assume symmetry but was incorrect, or there was a rare violation of destination based routing that affected accuracy; (2) The path returned by Reverse Traceroute is correct but incomplete (*e.g.,* private IP addresses in the middle of the path); (3) The path returned by Reverse Traceroute is correct and complete but is different from the direct traceroute path because of load balancing, path change, or IP addresses that do not allow for alias resolution; or (4) The direct traceroute is wrong (*e.g.,* in presence of per-packet load balancing). We now show that REVTR 2.0 almost never falls into case (1) and returns a wrong path (1.5% of paths when considering AS-level paths), whereas it is more likely to happen for REVTR 1.0 (8.3% of paths); and that the main cause of discrepancies between REVTR 2.0 and direct traceroute paths is the incompleteness of alias resolution techniques.

AS-level accuracy: The REVTR 2.0 AS line of Figure 5a shows that 5,246 of the 5,682 (92.3%) REVTR 2.0 reverse paths match the direct traceroute AS path, compared to 5,949 of the 7,275 (81.8%) for the REVTR 1.0 AS line. Of the 436 (7.7%) reverse traceroutes with discrepancies, 348 (6.1%) are cases when the reverse traceroute is incomplete, missing an AS hop, rather than wrong. Similar to how traceroute can return an incomplete path due to factors including unresponsive routers ("*") and hidden MPLS tunnels, Reverse Traceroute also can miss hops due to certain router configurations, such as routers that stamp RR packets with private IP addresses (that cannot be mapped to ASes) and routers that forward RR packets without stamping them. REVTR 2.0 can flag many of these measurements as potentially missing hops, even without access to the forward traceroute. Private IP addresses are directly seen in the IP level path, while we check for possible cases of routers that forward without stamping by translating the IP-level reverse traceroute into an AS level path and checking for suspicious AS links. We consider an AS link to be suspicious if the link is between a small AS s and a provider p of one of s's providers, and there is no known relationship between s and p in CAIDA's AS relationship dataset [58]. We consider an AS to be small if it has ≤ 5 providers and ≤ 10 ASes in its customer cone [58]. When a suspicious link is identified, REVTR 2.0 adds a "*" between the two hops, which occured on 10% of reverse traceroutes. For the 90% without "*", 98.3% (5,025 of 5,113) are correct and complete at the AS level.

For the remaining 88 (1.5%) REVTR 2.0 measurements with an AS path mismatch beyond a simple missing hop, it could be due to a route change in the interval between the reverse traceroute and direct traceroute measurements, or it could be an error. Errors can be caused by violations of destination-based routing, which Reverse Traceroute assumes holds. We checked for violations by replicating the methodology and updating the results of our prior work [34]. We find that only 6.6% of the 4,974,090 (hop, source) pairs that we test violate destination-based routing and only 1.1% of the (hop, source) pairs would affect REVTR 2.0 accuracy at AS level (Appx. E). For REVTR 1.0, of the 1,326 (18.2%) reverse AS paths that do not match the direct AS path, 604 (8.3%) have an interdomain symmetry assumption and an AS that does not appear in the direct AS path, strongly suggesting that those paths are wrong.

Router-level accuracy: Our analysis above suggests that 1.5% of REVTR 2.0 measurements have errors in their AS-level paths beyond a simple missing hop, but it is possible that more measurements have errors at finer granularities. Evaluating accuracy at router level is more difficult due to load balancing and incomplete alias information: 75% of the direct traceroute hops not seen in REVTR 2.0 paths do not allow for alias resolution, along with 81% of the extra hops seen by REVTR 2.0 that do not appear in the direct traceroute path. To corroborate that REVTR 2.0 does not return incorrect paths and that our ability to match routers seen in traceroutes is limited by missing alias information, we compute the same matching metric after eliminating REVTR 2.0 errors as a possible explanation for discrepancies: for each of our 8,093 source-destination pairs, we send an RR packet and a traceroute on the forward direction from the M-Lab sources to the RIPE Atlas probes, so that the path returned by RR is necessarily correct as it is extracted from a single packet. The *forward record route* lines show the 3,919 source-destination pairs where the RR packet reached the destination within 9 hops (so recorded the full path). We see that the forward record route lines are slightly below REVTR 2.0, with a median of 60% of traceroute routers also appearing in RR, as opposed to 67% for REVTR 2.0,

showing how hard it is to align RR and traceroute hops even when we know that the RR path is correct.

At the AS granularity, the forward record route line is similar to that of REVTR 2.0, showing that REVTR 2.0 achieves similar accuracy to a technique with a correct path. Finally, the *router optimistic* line shows the accuracy if all of the direct traceroutes hops that do not allow for alias resolution were actually on the reverse path, showing that the fraction of accurate reverse paths goes up from 25% to 68%. The real fraction of direct traceroute hops also seen in the reverse traceroute hops should lie in the shaded area, and it is probable that some of mismatches beyond the shaded region are due to traceroute and REVTR 2.0 measuring different valid paths in the presence of load balancing.

5.2.3 Coverage. We define the coverage as the percentage of reverse paths completely measured (*i.e.,* not aborted by REVTR 2.0 to avoid assuming interdomain symmetry) out of the number of reverse path measurements attempted.

Figure 5b shows the cost in coverage of making REVTR 2.0 more accurate than REVTR 1.0 (Fig. 5a): REVTR 2.0 can measure 6,322 (78.1%) of the reverse paths compared to 8,093 (100%) for REVTR 1.0. However, with REVTR 2.0, one can trust at least 98.5% of the reverse paths returned, whereas REVTR 1.0 gave no indication of which paths were likely accurate, eroding confidence in all paths returned, as at least 8.3% of the paths were wrong (§5.2.2). REVTR 2.0 still has solid coverage, as it can measure at least one reverse path from 39,544 out of 72,272 ASes (§5.1). Figure 5b demonstrates that adding timestamp measurements would only increase the number of successful reverse traceroutes by 85 (1%), even given perfect (unrealistic) information about adjacencies (Appx. D.1). To save probes and promote throughput, REVTR 2.0 does not use timestamp.

A key determinant of coverage is whether routers are responsive to and within range of record route. We update the results of our 2016 work [39], finding that record route responsiveness is similar to what we found in 2016 and much higher than had been previously thought. With REVTR 2.0's deployment, at least one vantage point is within 8 hops of (and hence able to measure reverse hops from) 63% of routers responsive to record route, compared to ≈ 12% in 2011 (Appx. F, Fig. 2 of RR paper [39]). The improvement is a result of the expansion of M-Lab and the flattening of the Internet [39]. We have spoken with the M-Lab team about future expansion to further improve this number.

5.2.4 Scalability. We evaluate three aspects of the scalability of the system: the throughput, the number of packets sent, and the time to measure a reverse traceroute. REVTR 2.0 can run 15M reverse traceroutes a day (§6.2), for a throughput of 173 per second. We calculate REVTR 1.0's throughput from our reimplementation (§5.2.1), which took 1,975 seconds to run 8,093 reverse traceroutes, *i.e.,* 354K a day or 4 per second. Table 4 shows the different types of packets sent by REVTR 1.0 and REVTR 2.0, along with the incremental benefits of the new components, namely:

$$\texttt{revtr 2.0} = \texttt{revtr 1.0 + ingress + cache - TS + RR atlas} \quad (1)$$

where `ingress` refers to our new technique to select record route vantage pionts based on their ingresses to the destination network

Type of packet	RR	Spoof RR	TS	Spoof TS	Total
REVTR 1.0	14,952	220,186	35,961	4,130	275,229
REVTR 1.0 + ingress	13,669	97,400	35,745	3,810	150,624
REVTR 1.0 + ingress + cache	12,708	64,310	35,765	3,925	116,708
REVTR 1.0 + ingress + cache − TS	12,690	64,435	0	0	77,125
REVTR 2.0 = REVTR 1.0 + ingress + cache − TS + RR atlas	11,831	61,080	0	0	72,911

Table 4: Number and type of packets sent, with incremental improvements. REVTR 2.0 sends 26% as many probes as REVTR 1.0.

(§4.3), `cache` corresponds to reusing traceroute and RR measurements for a day across multiple reverse traceroutes (Appx. D.2.2), `TS` stands for IP timestamp (which REVTR 2.0 no longer uses), and `RR atlas` refers to our new technique to increase the number of intersections between RR probes and the traceroute atlas (§4.2).

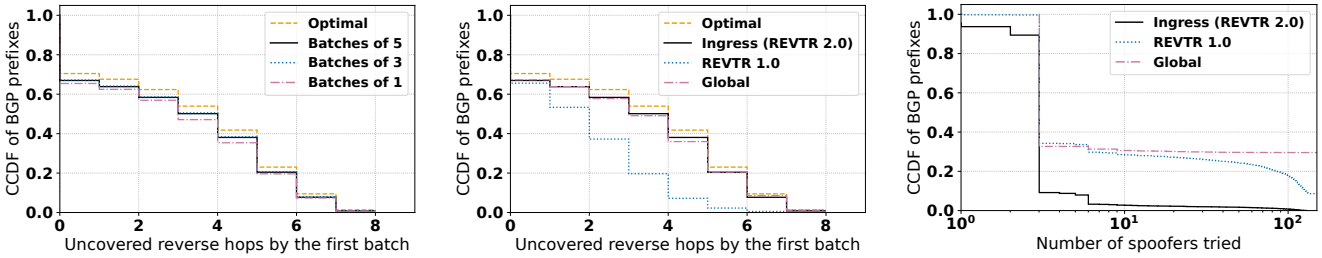
Overall, REVTR 2.0 sends 26% as many probes as REVTR 1.0 (73K vs 275K). Table 4 shows that most (125K) of the probe savings come from our new vantage point selection technique to send RR probes (§4.3). Each of the other components also contributes to the rest of the savings, namely: 34K for the cache, 39K for not using TS, and 4K for the RR atlas. The RR atlas does require sending RR packets, but the overhead is small compared to a long run of the system: REVTR 2.0 measures 15M reverse traceroutes in a day (§5.1), requiring 127M RR packets overall and 1M for the atlas technique. The traceroute intersections enabled by the RR atlas saved an additional 5.5% RR probes on our evaluation dataset, so we estimate it saved approximately 47M RR probes across the 101M reverse traceroutes of our large scale survey (§§ 5.1 and 6.2).

Figure 5c shows the CDF of run times for individual reverse traceroutes. The improvement between REVTR 1.0 and REVTR 2.0 is large: the median run time decreases from 78 seconds to 6 seconds. The improvement is due to the new record route vantage point selection technique, and the savings of the other parts of the system marginally reduce the time. The improvement is mainly due to how the spoofing implementation works: when the system sends spoofed packets, we set a timeout. This timeout must be set high enough to handle queuing delay on the vantage point and unpredictable latency. We empirically set this timeout to 10 seconds as only a few packets come back after the timeout and are therefore dropped, and each batch (§4.3) of spoofed packet incurs an additional 10 seconds to complete a reverse traceroute.

5.3 Picking RR vantage points

To evaluate REVTR 2.0's ingress-based technique for selecting RR vantage points (§4.3), we collect RR pings from our 146 M-Lab vantage points to 20 destinations in each of the 41,028 BGP prefixes seen in reverse or direct traceroutes (§5.2). We require each prefix to have at least three responsive destinations in the evaluation: two destinations to infer the ingresses needed by REVTR 2.0 to choose the vantage points, and a third to evaluate. Of the 41,028 prefixes, 26,169 satisfy this condition. Across these 26,169 prefixes, we have on average 9.6 responsive destinations, so the REVTR 1.0 set cover [52] is computed on a reasonable number of destinations.

Throughput: Two metrics impact the throughput of the system as a function of the batch size (number of vantage points attempted per measurement round): the number of batches tried before finding a reverse hop, and the number of reverse hops revealed by that batch



(a) 3 probe batches suffice. **(b) REVTR 2.0 finds more hops.** **(c) REVTR 2.0 tries fewer vantage points.**

Figure 6: Comparison of techniques to select record route vantage points.

	Fraction of BGP prefixes
Ingress	0.65
Ingress + double stamp	0.70
Ingress + double stamp + loop (REVTR 2.0)	0.71
REVTR 1.0	0.72
Optimal	0.72

Table 5: Fraction of BGP prefixes where techniques find a VP within 8 RR hops of the destination.

(both REVTR 1.0 and REVTR 2.0 stop whenever they find a reverse hop). Figure 6a shows the number of uncovered reverse hops by the first batch, for three different batch sizes. This parameter is crucial for scalability, as each batch adds 10 seconds to the reverse traceroute measurement time (§5.2.4). A batch size of 3 is a good compromise as increasing the batch size to 5 does not increase the number of discovered reverse hops further.

Figure 6b shows the CCDF of the number of uncovered reverse hops by the first batch (with a size of 3) by REVTR 1.0, REVTR 2.0, and by greedily trying the vantage points that are within range of the most prefixes globally (GLOBAL) [39]. The OPTIMAL line shows the result if we used the closest vantage point for each prefix, and its Y intercept shows that, for all techniques, the vast majority of destination prefixes in which the first batch uncovers no reverse hops are destinations for which no vantage point can measure reverse hops. REVTR 2.0 is nearly optimal and largely outperforms REVTR 1.0, which only discovers 4+ hops for 20% of prefixes compared to 50% of prefixes for REVTR 2.0.

Figure 6c shows the CCDF of the number of vantage points tried per BGP prefix for the three techniques before it either finds a reverse hop or gives up. REVTR 2.0 sends fewer probes. It tries 10+ vantage points for <5% of prefixes, versus 28% for REVTR 1.0 and GLOBAL. It tries 100+ vantage points for <1% of prefixes, versus at least 18% for REVTR 1.0 and GLOBAL.

Coverage: Table 5 indicates that 72% of prefixes had a vantage point in range (OPTIMAL). REVTR 2.0 found one for 71%.

6 EXAMPLE USE CASES

REVTR 2.0 meets the needs of our goal use cases (§3): on-demand operational use (§6.1) and large-scale topology mapping (§6.2).

6.1 Supporting traffic engineering

Context and motivation: Large providers like Microsoft, Google, and Facebook anycast their announcements from PoPs all around the world to thousands of peers [20, 26, 35, 80, 82]. This implies that

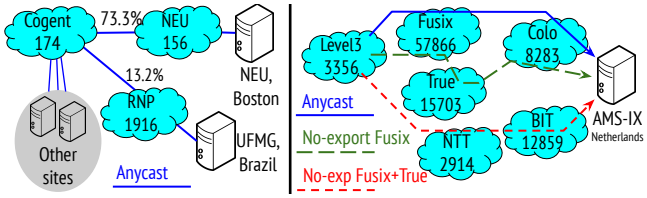


Figure 7: Ingress traffic engineering using PEERING and REVTR 2.0, shifting suboptimal Cogent routes toward US sites (left) and more evenly distributing routes across PEERING's AMS-IX providers (right).

where traffic enters their networks and which sites serve clients depend on the routing decisions of other networks. The performance, availability, and load on these services depend on how clients route to them. Providers have both limited visibility into and control over these decisions [57, 74, 77]. From speaking to operators at these companies in 2022, we know that some still lack visibility into routes to their prefixes. REVTR 2.0 provides a robust basis for traffic engineering (TE), yielding heretofore impossible precision to decisions on how to manipulate routing to achieve TE goals.

Methodology: We anycast a prefix from 7 PEERING locations [67] and choose destinations to monitor using an approach similar to proposals for how CDNs can choose representative monitoring targets [44, 82]: We group prefixes that have similar routing (*normalized routing state distance* less than 0.02 [45]) and pick the 15,300 groups with the most Google/M-Lab Speed Test measurements as a coarse measure of user activity, since Akamai argued that 15,300 targets can represent its clients in a recent study [82]. A contact at a global content provider reported that the chosen groups represent 70% of demand at the content provider. Within each group, we pick the prefix with the most Speed Test measurements. We then use as destinations all ingresses into these prefixes (§4.3), in order to follow CDN practice and probe routers at key convergence points, rather than end hosts. We deploy a REVTR 2.0 source on the anycast PEERING prefix and use REVTR 2.0 to measure the resulting routes.

Steering users to different sites: We first consider the case of a CDN attempting to group clients and steer them to different sites in an attempt to improve performance [35, 44, 82]. Figure 7 (left) considers an example of TE focusing on routes to PEERING traversing Cogent. REVTR 2.0 reveals that Cogent chose routes toward Northeastern University (NEU) in Boston (73.3% of reverse traceroutes through Cogent), UFMG in Brazil (13.2%), and other

sites. Cogent routers in the southeastern US chose routes to Brazil, which inflated latency for Cogent clients such as Leaseweb and the North Texas GigaPoP. We attempted to address this by poisoning Cogent on the announcement from UFMG to force Cogent to choose other sites [53]. Using Cogent's BGP communities to manipulate route preferences instead of poisoning would also suffice [8]. A second round of reverse traceroutes after the poisoning confirmed that 86.5% of routes through Cogent went to the PEERING site at NEU, and that RTTs to Leaseweb improved by 70ms and to North Texas GigaPoP by 99ms. Without access to REVTR 2.0, we would not have known how to manipulate advertisements to improve performance. For example, the *forward* path to North Texas GigaPoP is direct and does not traverse Cogent, and so traceroute would not have identified the AS causing the inflation.

Balancing load between providers: Figure 7 (right) considers a TE example focusing on balancing catchments between PEERING's two providers at AMS-IX, Coloclue and BIT. In the default anycast announcement, REVTR 2.0 reveals that 91.2% of destinations use Coloclue and 8.8% use BIT. Most of the networks reached Coloclue through Fusix, a Dutch transit provider. We used Coloclue's no-export large BGP community [5] to tell Coloclue not to export our prefix to Fusix. A second round of reverse traceroutes revealed that Fusix chose a route through True, another Dutch provider, which itself routed through Coloclue, and most networks continued to route via Fusix. For example, most routes through Level3 used Fusix in both configurations. We updated the TE to add Coloclue's no-export communities for both Fusix and True. A third round of measurements revealed that Fusix had chosen a route through Telia (not shown), which it probably did not export to Level3. Level3 routes shifted to NTT and BIT, achieving a more even 60.5%:39.5% split of destinations between Coloclue and BIT, as intended. Although Fusix's route now traverses a tier-1 network, the RTT between routers in Fusix and PEERING's AMS-IX site did not change significantly and remained below 2ms, indicating that Fusix clients would not experience performance degradation if a CDN performed similar TE. This sort of targeted TE is challenging without the visibility provided by REVTR 2.0, leading Google to conclude that it "needs [and lacked] the ability to gather information about the reverse path back from clients to Google's nodes" to debug performance problems [56] and causing Akamai to limit a recent TE system to coarse route optimization [82].

REVTR 2.0 supports quick iteration: Reverse traceroute measurements took 9–13 minutes per routing configuration, similar to the other required steps: 15 minutes to wait for BGP convergence and avoid route flap dampening [36, 41, 53, 71, 82] and 15 minutes to refresh the atlas. REVTR 2.0 can be integrated into automated TE systems, providing scale and coverage that cannot be matched by querying operators or looking glasses.

REVTR 2.0 in other TE work: In parallel work [85], we used REVTR 2.0 to debug new approaches we suggested for CDN route announcements that better balance control (directing users to particular sites) and availability (fast failover following site failure). We prototyped the approaches on our PEERING testbed [67] and used REVTR 2.0 to investigate instances of poor control, measuring the reverse paths from the client networks back to the PEERING sites when they were directed to undesired sites. REVTR 2.0 provided valuable information for troubleshooting routing configurations and poor performance that would traditionally be unavailable to CDNs [82].

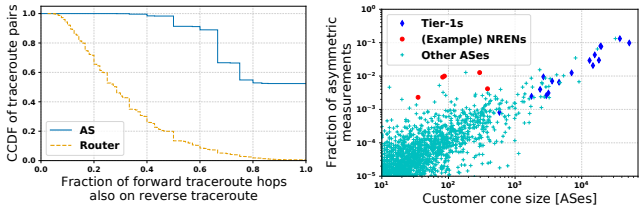


Figure 8: (a) 47% of paths are asymmetric at AS granularity, 99% at router. (b) Fraction of asymmetric routes traversing an AS vs customer cone size.

6.2 Measuring asymmetry at scale

Route asymmetry can lead to differences in one-way latency [31, 65], which degrade clock synchronization mechanisms [46], Internet latency-based positioning systems [78], and interactive applications such as gaming and videoconferencing. Route asymmetry also complicates failure and performance troubleshooting [51, 53]. Unfortunately, our understanding of routing asymmetry in the Internet is limited due to our previous inability to measure reverse paths at scale. We use our bidirectional measurement campaign (§5.1) to perform a study of path asymmetry in the Internet of unprecedented scale. We report on asymmetry over 30M paths for which we measured complete forward and reverse paths, where prior work had considered 120K reverse paths using RIPE Atlas [30]. More importantly, REVTR 2.0 allows one to compute asymmetry between vantage points (including in the cloud) and *any* destination of interest. We quantify path *symmetry* as the fraction of hops on the forward traceroute that are also on the reverse traceroute.

Overall asymmetry: Figure 8(a) shows the CCDF of symmetry at different granularities. Paths exhibit significant asymmetry. Only 53% of paths are symmetric even at the coarse AS granularity (where the 'AS' line meets the $x = 1$ axis in Fig. 8(a)). Asymmetry is even greater at the router granulary, with half the reverse traceroutes including less than 28% of the routers in the forward traceroute (where the 'router' line crosses $y = 0.5$). Computing asymmetry at the router granularity is challenging due to load balancing and missing aliasing information (§5.2.2). Figure 5a showed that, in the median case, a reverse traceroute would miss 33% of the routers seen by a forward traceroute issued in the same direction, and so we would expect to (falsely) infer 33% asymmetry even if the paths were symmetric. Combining these, we estimate that the median reverse path could share up to 28% + 33% = 61% of the hops on the forward path, missing at least 39% of the hops due to asymmetry. We also find that paths are more often symmetric near sources and destinations than in the middle (Appx. G.2) and that our results uphold much smaller previous studies (Appx. G.3).

Asymmetry and the Internet hierarchy: Figure 8(b) shows a scatter plot of the fraction of measurements where an AS was part of an observed AS-level asymmetry as a function of the AS's customer cone size [58]. Large transit providers appear on many AS paths, and the relationship between customer cone sizes and asymmetry indicates that large networks are also frequently involved in

asymmetric routing, with tier-1 networks occurring on many asymmetric paths. Some networks with few customers have a disproportionately large presence in asymmetric routes, *i.e.,* ASes towards the top left of Figure 8(b). Manual inspection of these networks reveals national research and education networks (NRENs) and regional providers. The red circles are four NRENs: Brazil's National Network for Research and Education (AS1916), Japan's National Institute of Informatics (AS2907), the Greek Research and Technology Network (AS5408), and GEANT (AS20965 and AS21320). NRENs show up frequently on forward traceroutes from a few M-Lab nodes hosted on education and research institutions (which get transit from NRENs, known to peer widely and employ a multi-AS "cold potato" policy), but reverse traceroutes from remote destinations often reach M-Lab nodes in education and research institutions through a regional commercial ISP. For example, traceroutes from mnl01 in the Philippines go via Japan's National Institute for Informatics (AS2907) to NTT, while reverse traceroutes from destinations reach mnl01 through Converge (AS17754), a regional ISP.

6.3 Other use cases

A security company has been using the demo version of our Reverse Traceroute system [12] to identify hidden providers on reverse paths to facilitate takedown of malicious activity. We expect other uses to emerge as we open the system, *e.g.,* using bidirectional path information in tomography measurements [52, 84].

7 RELATED WORK

All the related work on *measurement techniques* [37, 51, 69, 70], *techniques for inferring reverse path information* [51, 61, 62, 64, 81], and *systems that would benefit from reverse path information* [27, 32, 42, 50, 59, 62, 79, 83] cited in the original Reverse Traceroute paper [52] still benefit from this work. We focus here on the related work that has been published since then.

Internet scale traceroute techniques: Techniques exist for fast forward path measurements at Internet scale [23, 48, 76]. However, these techniques do not work for measuring reverse paths because building a reverse path is done incrementally.

Making efficient use of constrained vantage points: Multiple techniques deal with budget-limited measurements and minimize route measurement cost to gather the most information within budget constraints. Systems such as DTRACK [28], Sybil [29], and SIGNALS [38] attempt to track path changes to minimize staleness. We limit the impact of staleness on REVTR 2.0, but REVTR 2.0 could adopt such techniques to maintain even more up-to-date measurements. Other solutions reduce probing cost to gather additional information with traceroute measurements (*e.g.,* load balancers [18, 40, 75] or subnet topologies [24]). Our work similarly aims to reduce probing costs. Our technique to find ingresses to select the record route vantage points (§4.3) is similar to Rocketfuel's ingress reduction technique [72]. However, the goal is different, as REVTR 2.0 tries to quickly identify the closest vantage point, whereas Rocketfuel tries to reduce redundancy and treats each vantage point using the same ingress as equivalent.

Route asymmetry: Our work extends much smaller studies of path asymmetry [30, 47, 52]. Appendix G.3 compares our findings to the most recent characterization [30].

8 ETHICS

This work does not raise any ethical issues, although the active measurements and spoofing aspects of the system may raise concerns. The probing rate is controlled to not overload the network, limited to 100 pps per vantage point, a rate that has been shown to not trigger rate limiting on most networks [39]. Spoofing is controlled in Reverse Traceroute: the spoofed addresses are addresses of machines that participate in our system and have explicitly allowed our system to spoof their addresses.

9 CONCLUSION

We presented REVTR 2.0, the first system capable of measuring reverse paths at Internet scale. Focusing on the goals of accuracy, coverage and throughput, we designed a system that made principled tradeoffs to enable accurate measurement of 15M reverse paths in a day across 40K ASes, multiple orders of magnitude more than any prior work. REVTR 2.0 allows us to conduct Internet-scale topology measurements, such as our study confirming that most paths are asymmetric, and we demonstrated how operators can use it to support traffic engineering, load balancing, and troubleshooting based on knowledge of bidirectional paths.

REVTR 2.0 as a service: We run an instance of REVTR 2.0 as a service to the community, open to researchers and network operators. Users can access our service and its distributed vantage points to run reverse traceroute measurements from destinations they specify to M-Lab sources or to their own sources. Those who wish to measure reverse paths toward sources in the system can do so via the website, or via REST and gRPC APIs. Details on the implementation and how to join the system, run measurements, and add a source are given in Appendix A.

ACKNOWLEDGEMENTS

This work was partially funded by NSF Grants CNS-0905568, CNS-1405871, CNS-1836872, and CNS-1835252; and by CNPq project 432339. We thank the M-Lab team for providing the infrastructure and support essential to REVTR 2.0, especially Nathan Kinkade and Stephen Soltesz who are enabling the evolution of REVTR 2.0 on the M-Lab infrastructure. Our system builds on the work earlier team members of the Reverse Traceroute team over the last decade+, including the co-authors of the 2010 Reverse Traceroute paper; Rob Hansen, who built the Go version of REVTR 1.0; Kevin Lee, who helped deploy REVTR 2.0 on M-Lab; Alex Stein, who started the work for vantage points selection; and Muzammil Abdul Rehman, maintained earlier versions of the system. A special thanks to; Ralph Holz, who provided valuable feedback on our revisions; Phillipa Gill who invited us at the M-Lab tutorial at SIGCOMM 2022 to present this work; the RIPE Atlas people for providing measurement credits to support this work; and to our shepherd Rachee Singh and the anonymous reviewers for their careful reading of this paper and suggestions for its improvement. Kevin Vermeulen first started this work when he was a postdoc at Columbia University and pursued it as a CNRS researcher at LAAS-CNRS, Universite de Toulouse, CNRS, Toulouse, France.

APPENDICES

A OUR OPEN REVTR 2.0 SYSTEM

An open system and open source code: We operate REVTR 2.0 as an open system: researchers, operators, and others can sign up as users of our system, optionally add their own hosts as Reverse Traceroute sources [15], and request reverse traceroute measurements from destinations of their choice to their sources or our existing sources. Our system orchestrates the distributed vantage points to implement the REVTR 2.0 approach to service the request, and it stores the resulting reverse traceroutes as well as returns them to the requesting users.

Since our system supports users adding their own sources and requesting arbitrary measurements to them, users can measure paths of interest without the complexity of deploying their own instance. We encourage people to use our REVTR 2.0 system rather than running their own instance, because REVTR 2.0 needs access to and careful coordination of vantage points worldwide with specific and somewhat rare requirements, such as the ability to send spoofed packets. We do, however, make our code available [14]. It is written in Go and employs a service-oriented architecture, decisions that helped us achieve maintainability, efficient use of resources, and scalability.

System architecture and implementation: Our system consists of distributed sources, a subset of which also serve as record route vantage points, and a number of centralized services that coordinate the sources and vantage points, enact the REVTR 2.0 measurement logic, request and cache background measurements such as the traceroute atlas, store measurement results, and so on. Each centralized service is packaged as a Docker container to simplify deployment and migration. They run on three servers at Northeastern University: a storage server, with 24 CPUs, 64 GB RAM and 3.6TB storage; a server that runs the service to select vantage points to issue spoofed record route probes (§4.3), with 32 CPUs and 384 GB of RAM, and a server running all other services, with 8 CPUs and 64 GB RAM.

Currently, REVTR 2.0 record route vantage points are deployed at all M-Lab sites (except the small number that are hosted on networks that do not allow spoofing or that filter record route packets).

Adding users and sources: To use our system, researchers and operators request to be added as users in our user database (currently maintained manually). The database includes two per-user rate limiting parameters that we use to prevent a user from overloading the system: the number of parallel reverse traceroute measurements that the user can request, and the maximum number of reverse traceroute measurements that the user can issue per day, similar to what RIPE Atlas does [16].

To gain the ability to measure reverse traceroute to their own hosts, users add them as sources to our system. To add a source, the user makes a request, and we manually add it to a database listing approved sources. The user can configure whether the source serves as a record route vantage point to aid in servicing Reverse Traceroute requests, an option that we disable by default. We supply a Docker container that the user deploys on the source. The container automatically connects to our source controller service. The controller authenticates the source based on the database and runs a bootstrap process that starts by checking whether the source can receive record route packets. If successful, the process then requests traceroutes from REVTR 2.0 vantage points and RIPE Atlas to the source to build its traceroute atlas (Q1), as well as record route probes to the traceroute hops to aid in identifying intersections (Q2). The bootstrapping process takes around 15 minutes, most of it incurred by running traceroutes on RIPE Atlas. RIPE has kindly provided us with RIPE Atlas credits to support REVTR 2.0. If a user wants frequent refreshing of the source's traceroute atlas, the user provides additional credits by giving us a RIPE Atlas key that we associate with the source.

Issuing and storing Reverse Traceroute measurements: After a user is in our database, the user can request Reverse Traceroute measurements via either REST or gRPC APIs from any destination to the sources registered in the system. The user can specify options to tune the request, such as how stale traceroutes are allowed to be and whether to run a forward traceroute after the Reverse Traceroute completes. Our documentation provides details [14].

In addition to these on-demand user-driven measurements, we have a service running on the M-Lab nodes where the NDT service is running. NDT is best known as the service backing Google's `Internet speed test` and measures performance between a client and an M-Lab server. When a client initiates an NDT measurement, our service requests a reverse traceroute to measure the route from the client to the same M-Lab server. Whether REVTR 2.0 accepts or rejects the request depends on system load. M-Lab already issues a forward traceroute from M-Lab to the client, which our measurements complement. Over time, these measurements will constitute a dataset providing round-trip views of paths with the NDT-measured throughput and latency achieved over those paths.

Our system archives both user-driven and NDT-based reverse traceroutes to M-Lab's Google Cloud storage [13].

B MAPPING IP PATHS TO COARSER GRANULARITIES

B.1 Alias resolution

Alias resolution, the process of grouping IP addresses into routers, is an open problem. It is used by several sections in this paper: the traceroute atlas (Q1, Q2), the ingress technique (Q3), the comparison with REVTR 1.0 (§5.2), and the use cases (§6). In all of these sections, we use as a basis the CAIDA ITDK alias dataset [6] that is only based on MIDAR [55] to minimize the number of false positives and the SNMPv3 technique consisting of extracting router identifiers from responses to unsolicited SNMPv3 requests [17]. In addition, for certain sections, we use ad hoc heuristics to perform aliasing tailored by the use case and the data available in that section.

In the traceroute atlas: Our new technique for increasing the set of intersections of the atlas (Q2) faces the non trivial problem of matching record route hops with traceroute hops. We use two techniques to find correspondence between a RR hop and a traceroute hop. First, we run MIDAR [55] on the set of RR hop and traceroute hops. A significant portion of the IP addresses (30%) revealed in the RR measurements do not appear in ITDK, so running our own

MIDAR improves the alias coverage. Second, we look for point-to-point links, which we identify as two IP addresses being in the same /30 or /31 prefix, as this is a common subnetting practice followed by network operators [43]. With the assumption of traceroute hops revealing ingress IP addresses (a classic but non-standard behavior [73]) and RR hops revealing egress interfaces [4], a point to point link is composed of a RR hop *followed by* a traceroute hop. The same technique is used in evaluation when comparing reverse paths returned by REVTR 1.0 and REVTR 2.0 to direct traceroute paths, and in the asymmetry survey (§6.2) to compare forward paths and reverse paths.

B.2 IP to AS mapping

As for the alias resolution problem, the IP to AS mapping is still an active area of research, and the better it is, the more accurate our results are. There are several sections that use IP to AS mapping: when REVTR 2.0 relies on IP to AS mapping to decide whether it aborts the reverse traceroute measurement if it has to assume symmetry on an interdomain link (§4.4); throughout the evaluation section (§5) and the use cases section (§6) to get AS level paths from reverse paths measured by REVTR 2.0.

We adopt the following method to map IP addresses to AS, borrowed from Arnold *et al.* [20]: for each hop in a reverse path, we prioritize IP to AS mapping from EuroIX [9] over PeeringDB [11] over RouteViews [3] over Whois. We cannot directly use BDRMAPIT [63] for several reasons: BDRMAPIT is an offline tool, that takes a set of traceroutes in input to build a topology graph and infer IP to AS mapping from it. When measuring a reverse path, one does not know the hops in advance except the hops in the traceroute atlas. Even if we knew the hops in advance, BDRMAPIT expects traceroutes, and our measurements include hops measured with RR that may violate the properties that BDRMAPIT expects. Specifically, RR paths can not be easily transformed into traceroute paths [70] precisely because of the presence of routers stamping record route packets with private IP addresses or not stamping at all (§5.2.2).

The only case where BDRMAPIT could change REVTR 2.0's results is when REVTR 2.0 issues a forward traceroute to the current hop and assumes symmetry between the penultimate hop of the traceroute and the current hop (Q5): if the penultimate hop intersects a traceroute of the atlas, the penultimate hop is a hop given as input to BDRMAPIT, and its IP to AS mapping could change between our technique and BDRMAPIT. If its IP to AS mapping changes and the link between the penultimate hop and the current hop switches from interdomain to intradomain or conversely, it would change the decision of REVTR 2.0 to abort or continue to measure the reverse path. Changing from intradomain to interdomain is problematic for accuracy, as REVTR 2.0 considers this symmetry assumption as being correct whereas it should not, and changing from interdomain to intradomain is problematic for coverage, as REVTR 2.0 should have considered the symmetry assumption as being correct (§4.4).

Evaluation of using BDRMAPIT: We run BDRMAPIT on the traceroute atlas, and for each symmetry assumption in the reverse path, we check if the link on which we assumed symmetry changes from intradomain to interdomain or vice versa.

Although BDRMAPIT changed the IP to AS mapping of 3.5K of 55K (6.8%) IP addresses of the traceroute atlas, we find that of the 102.6M symmetry assumptions, only 70K (0.07%) assumptions would change from intradomain to interdomain and 1.5M (1.5%) would change from interdomain to intradomain. The 70K intradomain assumptions becoming interdomain would only decrease the number of trustworthy paths (31M without interdomain symmetry assumption) by 8K (0.03%), and the 1.5M interdomain assumptions becoming intradomain would improve the number of trustworthy paths by 312K (0.1%). These results show the very limited benefit of BDRMAPIT over our technique for improving REVTR 2.0 accuracy and coverage, even when considering all BDRMAPIT inferences are correct. This is expected, as the only case where BDRMAPIT could change the type of a link affecting REVTR 2.0 accuracy and coverage is when REVTR 2.0 assumed symmetry on a link where the near side (the closer to the source) of the link is an intersection hop of the atlas, which is only a small fraction of the symmetry assumptions (7.3%). We note that incorrect BDRMAPIT inferences could negatively impact REVTR 2.0's accuracy as reverse traceroutes with interdomain assumptions of symmetry (incorrectly inferred to be intradomain by BDRMAPIT) would be considered safe. Similarly, incorrect BDRMAPIT inferences could also negatively impact REVTR 2.0's coverage as reverse traceroutes with intradomain assumptions of symmetry (incorrectly inferred to be interdomain by BDRMAPIT) would be discarded.

C HANDLING NON-STAMPING DESTINATIONS

Sometimes, the destination is configured to not stamp record route packets or to stamp them with a different IP alias that is not in the BGP prefix, so that our approach described in Section 4.3 fails to find any ingress. We developed two heuristics to identify such destinations and adapt our ingress identification for them.

Double stamp: The same IP address can be stamped in two adjacent record route entries, without the destination IP address appearing in the path, in two cases: (1) the IP address is an alias of the destination; or (2) the destination does not stamp record route packets, and the IP that we observe is the penultimate hop before the destination and is on both forward and reverse paths. To handle both cases, we consider the double stamped IP address to be part of the destination prefix for the purposes of ingress identification.

Loop: If there is a *loop* in a record route measurement (a pattern $a - S - a$ where a is an IP address and S is a loop-free subpath), it most likely indicates that the packet reached the destination, but the destination router did not record an address (or recorded an alias other than the specified destination), and a appears on both the forward and reverse path. In this case, we can not pinpoint where in S the destination was, and so we consider all IP addresses in S as candidate ingresses if they also appear on the RR probe from the same vantage point to the second destination in the prefix.

D COMPLEMENTARY DETAILS ON THE EVALUATION

This section gives more details on how we obtain the main results described in Section 5 about IP timestamp and maintaining the traceroute atlas.

D.1 Utility of IP Timestamp

This section justifies in detail why REVTR 2.0 does not use TS for large scale surveys. There are two reasons why the timestamp technique can fail: (1) either the system does not have the right adjacencies or (2) hops do not respond to TS. To establish if we are in case (1) or (2), we run REVTR 2.0 without TS (second-to-last line of Table 4), identify the first hop h of each reverse traceroute before the first hop assumed symmetric, and give h's true next hop from the ground truth traceroute as an adjacency. To find the right adjacency, we use the same aliasing technique as the one described in §5.2.2. If we are in case (1), TS probes will be able to find the adjacency, increasing REVTR 2.0 coverage.

Of the 1,771 (17.7%) reverse traceroutes where we assumed symmetry on an interdomain link, timestamps succeeded 221 (2.2%) times in finding the next hop, and in 85 (0.9%) of them REVTR 2.0 did not assume symmetry on an interdomain link once allowed to use timestamps. For these 85 reverse traceroutes, we look at how it affects accuracy, by comparing the hops discovered with the version without TS. We found that for 37 reverse traceroutes, the hop after the TS hop was not present in the reverse path without TS, and that for 17 of them, the AS of the hop after the TS hop was not in the AS path of the reverse path without TS, whereas it was present in the ground truth traceroute, showing that the reverse path without TS was certainly wrong. This study shows the tradeoff between issuing many more TS probes in the network versus the added value in coverage/accuracy, in the optimal case where the system is fed with the exactly right adjacencies. As we expect TS to perform even worse with incomplete/inaccurate adjacencies, we decide to not run them in REVTR 2.0 and instead reduce the probing overhead.

D.2 Building a traceroute atlas

D.2.1 Picking traceroutes. We perform a study that justifies our traceroute atlas design choice (§4.1), *i.e.,* randomly selecting 1000 traceroutes per source achieves nearly optimal performance. To this end, we collect traceroutes to 10 of our M-Lab sources from all the RIPE Atlas probes available at the time of measurements (> 10,000) on 04/01/2021. The experiment consists of using a portion of the traceroutes to simulate the atlas, while using others to simulate reverse traceroutes. To evaluate how selecting traceroutes at random performs, we assume that we have an oracle that knows all traceroute paths. We set as the optimal set of traceroutes the greedy selection of traceroutes from a weighted maximum coverage of IP addresses seen in the traceroutes, where the weight of an IP address is the sum of the distance to the source on each traceroute where it appears. The optimal selection gives us an upper bound to evaluate the performance of a random selection.

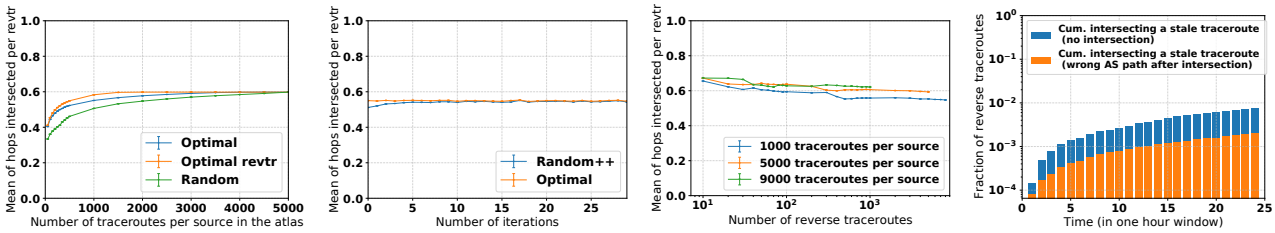
Figure 9a shows the results of random versus optimal on the following experiment: For each M-Lab node, we randomly split the 10,000 traceroutes in two subsets, 5,000 traceroutes that can be selected for the atlas, and 5,000 traceroutes corresponding to reverse traceroutes. The x axis represents the number of traceroutes that we allow for the atlas, and the y axis corresponds to the mean fraction of hops intersected per reverse traceroute. The optimal lines correspond to the selection described above, with a difference in the weights used to select the traceroutes: the *Optimal* line uses the weights of the atlas, while the *Optimal revtr* line uses the weights of the reverse traceroutes. There are two main lessons from this graph: first, only a small fraction of traceroutes is required to intersect most of the hops: On the *Optimal* line, 20% of the traceroutes (1,000 per source) gives a mean of 56% of the hops intersected per reverse traceroute, while 100% of the traceroutes (5,000 per source) increases the mean to just 60%. Second, random selection achieves near optimal performance: for 1,000 traceroutes per source, the mean is 50% compared to 56% for optimal. Figure 9b shows how many iterations the random atlas needs to converge towards the optimal one. The experiment consists of initializing the atlas with 1,000 random traceroutes per source (among the 5,000 traceroutes available for the atlas). Then, an iteration consists of simulating 1,000 random reverse traceroutes (among the 5,000 traceroutes available for the reverse traceroutes). Then, between each iteration, we only keep in the atlas the minimal set of traceroutes to cover the intersections used by those reverse traceroutes, and replacing the others with new ones to reach 1,000 total traceroutes per source. We find that five iterations suffice so that the mean fraction of hops intersected for the random atlas converge towards the value of the optimal atlas. In other words, as we refresh the atlas every day, five days are enough to obtain an optimal atlas.

Fig. 9c shows a different view of the data, to see if the mean fraction of hops intersected with 1,000 traceroutes per source holds when the number of reverse traceroutes increases. To evaluate that, we vary the number of reverse traceroutes and observe how the mean fraction of intersected hops per reverse traceroute evolves. Each line shows a fixed size of the traceroute atlas. We observe that the mean decreases very slowly, with a < 1% decrease between 1,000 and 9,000 reverse traceroutes, so we conjecture that this trend will hold true for a larger number of reverse traceroutes (*i.e.,* millions). We also ran the same study using 10 RIPE Atlas probes as sources and found similar results, showing that the properties are not M-Lab specific.

D.2.2 When to refresh measurements? We evaluate our strategy to refresh measurements every day. We use REVTR 2.0 for 24 hours to measure all paths between 146 sources and destinations taken from the ISI hitlist [33] with a fresh atlas. To identify if traceroutes are stale and their impact on accuracy, we enqueue a new measurement of any atlas traceroute intersected by a reverse traceroute. Every 15 minutes, we run queued traceroutes in a batch[1]. Our analysis compares the aging traceroutes in the atlas (intersected by the reverse traceroutes) with the fresh traceroute measurements. As a result, the number of missed changes is an approximation of the real number, as changes could have happened between the time the reverse traceroute intersected and the time RIPE atlas performed the measurement.

Figure 9d shows stacked bars of the cumulative number of reverse traceroutes that intersected a stale traceroute. We say a traceroute is stale if either the intersection does not exist in the fresh traceroute or the AS path after the intersection changed. The first case is conservative as the change might occur before the intersection

[1]We choose this approach rather than directly running the traceroute because we are limited by how the number of parallel measurements is counted and limited on the RIPE Atlas API.



(a) Savings vs traceroutes in atlas (b) Convergence to optimal (c) Savings vs # of revtrs (d) Impact of staleness

Figure 9: Mean fraction of hops saved depending on the number of traceroutes in the atlas, for Optimal and Random selection (a). Mean fraction of hops saved depending on the number of traceroute atlas replacement policy iterations (b). Mean fraction of hops saved depending on the number of reverse traceroutes to intersect (c). Fraction of reverse traceroutes intersecting a stale traceroute over a day (d).

and the subpath between the intersection and the source still be valid. After 24 hours, *only 0.7% of reverse traceroutes intersected a stale traceroute.* To further reduce this number, two options can be combined: reduce the size of the atlas and refresh it more frequently, or use more accurate techniques that capture staleness [38] and refresh the traceroutes as they become stale.

E VIOLATION OF DESTINATION BASED ROUTING

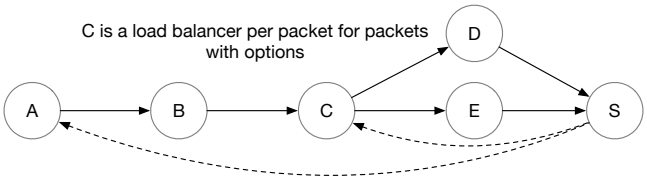


Figure 10: Reverse Traceroute reveals one accurate reverse path in presence of load balancing.

During May 2022, we partly replicated a 2012 study quantifying the violations of destination based routing [34] to show that the violations affecting REVTR 2.0 accuracy are rare. To check for violations, we perform spoofed record route pings to the 57M destination of our survey (§6.2), spoofed as each of our 146 M-Lab sources. For each ping that uncovers at least two reverse hops on the reverse path between a destination D and a source S, for each pair of adjacent reverse hops (R, R'), we check whether a spoofed RR ping to R spoofed as S also goes through R'. If it does not, we say that R violates destination based routing, as it does not always use the same route to destination S.

We are looking for violations that can affect REVTR 2.0 accuracy, and so we want to exclude cases attributable to load balancing, in which case REVTR 2.0 returns a single valid path among the load balanced paths (Fig. 10). Whereas load balancers can induce false links in forward traceroutes because each hop is measured by a different packet which may be load balanced differently [21], a link measured by record route for Reverse Traceroute is traversed by a single packet that records the hops on both sides of the link. To minimize false links caused by the traceroute atlas, REVTR 2.0 uses the Paris traceroute technique [21] to measure a single consistent path in the face of per flow load balancing. Load balancers balance packets with options randomly (as opposed to per flow), and so we identify violations caused by load balancing by sending multiple (spoofed) packets to R [34]. If the measurements return multiple next hops, we classify R as a load balancer and do not consider it as a problematic source of violations of destination-based load balancing.

Of the 4,974,090 tuples (R, R', S), 328,726 (6.6%) violate destination based routing and cannot be attributed to load balancing, potentially affecting REVTR 2.0 accuracy at router level. Of these 328,726 violations, 64,471 (1.3%) cause an AS path deviation, affecting REVTR 2.0 accuracy at AS level. If one wants to further improve REVTR 2.0 accuracy, we provide the possibility to perform these redundant measurements as part of reverse traceroute measurements to identify these violating routers, at the cost of extra measurements. The measurements could be flagged as suspicious, the different routes uncovered by the redundant measurements could all be returned, and/or the system could be configured to not perform RR measurements to these hops, either trying their predecessors or aborting the reverse traceroute.

	2016	2020
All probed	510,305	829,749
Ping responsive	394,644 (77%)	604,454 (73%)
RR responsive	296,734 (58%)	472, 043 (57%)
RR reachable in 8 or fewer hops	182,926 (36%)	298,078 (36%)

Table 6: A larger fraction of destinations that respond to pings (without IP options) also respond to pings with the record route option in 2020 (78%) in comparison to 2016 (75%, from [39])

F RECORD ROUTE RESPONSIVENESS AND REACHABILITY

Since Reverse Traceroute relies on the record route IP option to build accurate reverse paths, we assess the record route coverage

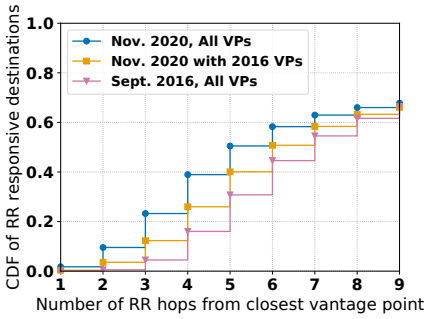


Figure 11: RR hops from the closest M-Lab vantage point to RR responsive destinations, in 2016 versus 2020. RR responsive destinations are closer to M-Lab vantage points in 2020, in comparison to 2016.

of the set of vantage points REVTR 2.0 uses. We are interested in assessing three key metrics that affect how well REVTR 2.0 can function: (1) whether destinations respond to probes with the record route IP option enabled, (2) whether destinations are within 8 record route hops (so we can uncover hops along the reverse path, as described in Section 2), and (3) how close the destinations are to their closest VP (so we can maximize the number of hops uncovered on the reverse path). In November 2020, we conducted a survey similar to one we conducted in September 2016 [39]. We selected one responsive destination per BGP prefix in ISI's hitlist [33] and sent one record route option-enabled ping, as well as three regular pings with no IP option enabled from each of our M-Lab vantage points to the selected hosts.

Table 6 summarizes key statistics from both September 2016 [39] and November 2020 measurement rounds. A host is considered:

- Ping responsive if it responds to a ping (with no IP option enabled).
- RR responsive if it responds to a record route enabled ping.
- RR reachable if we observe its IP address anywhere in the nine slot record route array.

Between 2016 and 2020, we do not see any significant changes in terms of overall RR responsiveness or RR reachability, and in both years 62-63% of RR-responsive destinations are within the 8 hops required by Reverse Traceroute. Spoofing allows Reverse Traceroute to use the closest vantage point to a destination [52], regardless of the source, and so REVTR 2.0 can use record route to measure reverse hops from 63% of RR-responsive destinations for all sources. Without this ability to spoof to decouple the forward and reverse paths and use the closest vantage point, record route could only be used if the particular source was within 8 hops of the destination, which occurs for only 32% of ⟨`source, RR-responsive destination`⟩ pairs. So spoofing nearly doubles the coverage REVTR 2.0 obtains with record route from 32% to 63%.

RR responsive destinations tend to be closer (in terms of record route hops) to M-Lab vantage points in 2020. Figure 11 shows the distribution of the distance between RR-responsive destinations and their closest VPs, at different times (2016 and 2020) and with different VP sets. The lines `Nov. 2020, All VPs` and `Sept. 2016, All VPs` both use all available M-Lab vantage points at the time of the experiment. We observe, for example, that 39% of responsive destinations are within 4 hops in 2020, compared to 16% in 2016.

We are also interested in measuring differences without the effects of the change in available VPs over the years. The line `Nov. 2020 with 2016 VPs` shows results restricted to the 44 2020 sites that were also among the 86 M-Lab sites in 2016. For both of the November 2020 lines, we only consider destinations that were RR-responsive to at least one of these 44 sites in order to have a consistent denominator (450,059 hosts). Even using only roughly half the sites that were available in 2016, destinations tend to be closer to the closest M-Lab site in 2020 (*e.g.,* RR reachability within 4 hops jumps from 16% with 86 sites in 2016 to 26% in 2020 using 44 of the sites).

These are positive results for REVTR 2.0—being within 8 hops allows for hop discovery on the reverse path (§2), which contributes to more accurate reverse traceroutes by avoiding assumptions of symmetry, and being closer than 8 hops enables measuring more hops per packet, reducing probing overhead and increasing system throughput. While the availability of new vantage points placed "in diverse location[s] around the world" and "in well connected data centers where ISPs interconnect" [10] is a factor, we suspect that increased peering could be an additional explanation as to why we observe shorter paths from the same vantage points, 4 years later.

G ADDITIONAL RESULTS ON PATH SYMMETRY

In this appendix we discuss complementary results to our analysis of path asymmetry in the Internet (§6.2) using the same dataset.

G.1 Impact of symmetry assumptions

Our path symmetry study considers 30M complete bidirectional measurements whose REVTR 2.0 did not make any interdomain symmetry assumptions. Our results have shown that intradomain symmetry assumptions do not have a significant impact on accuracy (§4.4). We now show that intradomain symmetry assumptions also do not significantly impact our path symmetry study.

Fig. 12 quantifies route symmetry considering the subset of 23M bidirectional measurements whose REVTR 2.0 does not make any symmetry assumptions. Results are qualitatively similar to those in Figure 8, with 3% more symmetric paths at the AS granularity and even smaller differences at the router granularity.

G.2 Asymmetry and AS-paths

We find that asymmetric routes are usually longer than symmetric ones. Fig. 13 shows the CDF of AS-path lengths for all measurements as well as for symmetric and asymmetric measurements traversing a Tier-1 network. Similarly, symmetric paths through Tier-1 networks are often shorter than asymmetric paths, with most paths traversing 5+ ASes being (rare and) asymmetric.

Fig. 14 shows where asymmetries are detected in forward traceroutes. We show, for different AS-path lengths, the probability that each hop on the forward traceroute is also on the corresponding reverse traceroute. We find that hops towards the middle of the path have a higher probability of being asymmetric, with a bias towards hops close to the M-Lab node (left side of Fig. 14) being missing

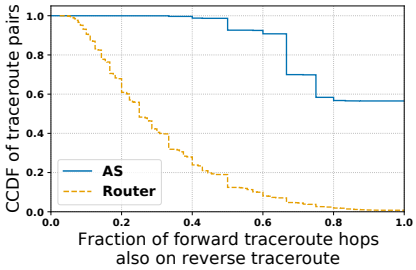


Figure 12: Symmetry for measurements whose reverse traceroute makes no symmetry assumptions, results are similar to Figure 8.

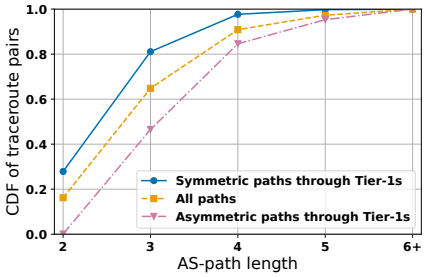


Figure 13: Distribution of AS-path lengths for different subsets of measurements. Symmetric paths are usually shorter than asymmetric ones.

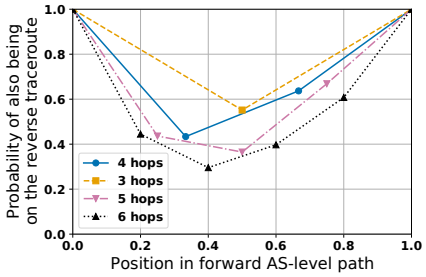


Figure 14: Probability that hops on a forward traceroute are also on the reverse traceroute as a function of hop distance from M-Lab.

from the reverse traceroute more often than hops close to the destination. Fig. 14 indirectly shows that instances of asymmetry often involve multiple hops, particularly on longer routes.

G.3 Comparison with previous work

Previous works have studied route asymmetry on the Internet [30, 47, 52]. With the exception of the original Reverse Traceroute paper [52], all previous works relied on standard traceroute and were thus limited to bidirectional measurements between a few hundred controlled vantage points. REVTR 2.0 allows us to cover a significantly wider range of destinations and cover a more diverse set of Internet paths.

More recently, de Vries *et al.* [30] characterized route asymmetry using 119,550 bidirectional route measurements between 4000 RIPE Atlas vantage points. A key difference in our study to theirs is in the very definition of asymmetry. In particular, de Vries *et al.* quantify asymmetry as the edit distance between the forward and reverse traceroutes, while we quantify symmetry as the fraction of hops on the forward traceroute also on the reverse traceroute. Our definition of asymmetry may underestimate the amount of actual asymmetry in Internet paths, which may explain why de Vries *et al.* found 87% of paths are asymmetric while we find only 47% of paths are asymmetric at the AS granularity.

Although Fig. 14 is inspired by and similar to Figure 7 in [30], the two figures differ not only in the definition of symmetry (above), but also in that we do not require that the forward and reverse paths have the same length. Compared to [30], our results show a concentration of asymmetry close to the source, which is expected since our definition of asymmetry only considers hops on the forward traceroute.

Finally, de Vries *et al.* found that tier-1s and large IXPs often show up on asymmetric paths (Table 2 in [30]). We find similar results in Figure 8(b), and name the top 10 ASes that appear in asymmetric routes in Table 7. Our results, however, do not identify IXPs as frequent occurrences in asymmetry routes as we map IXP addresses to the corresponding member AS using PeeringDB (Appx. B.2).

RANK	ASN	NAME	PREV.	TIER	CC	[30]
1	174	Cogent	0.132	2	32078	2
2	1299	Telia	0.132	1	37380	3
3	3356	Level3	0.097	1	51753	1
4	3257	Tinet	0.079	1	19143	4
5	2914	NTT	0.076	1	19604	8
6	6939	HE	0.064	2	17237	—
7	6453	Tata	0.043	1	15712	9
8	9002	RETN	0.034	2	8084	—
9	6762	Sparkle	0.029	1	18130	—
10	3491	PCCW	0.029	1	13127	—

Table 7: Top 10 ASes most frequently involved in path asymmetry. The 'prevalence' column shows the fraction of asymmetric paths where the AS is part of the observed asymmetry, the 'CC' column shows the AS's customer cone size, and the last column shows the rank of the network in [30].

REFERENCES

[1] CAIDA Ark. https://www.caida.org/projects/ark.
[2] iPlane Traceroute Dataset. http://web.eecs.umich.edu/~harshavm/iplane/iplane_logs/data/.
[3] Oregon Route Views. http://www.routeviews.org/.
[4] RFC 791, 2007. https://datatracker.ietf.org/doc/html/rfc791.
[5] RFC 8092, 2017. https://tools.ietf.org/html/rfc8092.
[6] CAIDA Ark IPv4 Internet Topology Data Kits Dataset, 2019. https://www.caida.org/catalog/datasets/internet-topology-data-kit/.
[7] Visible ASNs: Customer Populations Estimations, 2022. https://stats.labs.apnic.net/aspop/.
[8] OneStep—Cogent BGP Communities, 2022. https://onestep.net/communities/as174/.
[9] Euro IX, 2022. https://ixpdb.euro-ix.net/en/.
[10] M-Lab Status, 2022. https://www.measurementlab.net/status/.
[11] Peering DB, 2022. https://www.peeringdb.com/.
[12] Reverse Traceroute Demo, 2022. https://www.revtr.ccs.neu.edu.
[13] Reverse Traceroute MLab Data Archive, 2022. https://console.cloud.google.com/storage/browser/thirdparty-revtr-mlab-oti.
[14] Reverse Traceroute System Code, 2022. https://github.com/NEU-SNS/ReverseTraceroutePublic.
[15] Reverse Traceroute Vantage Point Code, 2022. https://github.com/NEU-SNS/revtrvp.
[16] RIPE Atlas, 2022. https://atlas.ripe.net/.
[17] Taha Albakour, Oliver Gasser, Robert Beverly, and Georgios Smaragdakis. Third Time's not a Charm: Exploiting SNMPv3 for Router Fingerprinting. In *Proc. ACM IMC*, 2021.
[18] Rafael LC Almeida, Italo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. Classification of Load Balancing in the Internet. In *Proc. IEEE INFOCOM*, 2020.
[19] Ruwaifa Anwar, Haseeb Niaz, David R. Choffnes, Italo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating Interdomain Routing Policies in the Wild. In *Proc. ACM IMC*, 2015.

[20] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. In *Proc. ACM IMC*, 2020.

[21] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proc. ACM IMC*, 2006.

[22] Adam Bender, Rob Sherwood, and Neil Spring. Fixing Ally's Growing Pains with Velocity Modeling. In *Proc. ACM IMC*, 2008.

[23] Robert Beverly. Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery. In *Proc. ACM IMC*, 2016.

[24] Robert Beverly, Arthur Berger, and Geoffrey G Xie. Primitives for Active Internet Topology Mapping: Toward High-frequency Characterization. In *Proc. ACM IMC*, 2010.

[25] Randy Bush, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Internet Optometry: Assessing the Broken Glasses in Internet Reachability. In *Proc. ACM IMC*, 2009.

[26] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. Analyzing the Performance of an Anycast CDN. In *Proc. ACM IMC*, 2015.

[27] Yan Chen, David Bindel, Hanhee Song, and Randy H Katz. An Algebraic Approach to Practical and Scalable Overlay Network Monitoring. In *Proc. ACM SIGCOMM*, 2004.

[28] Italo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. DTRACK: A System to Predict and Track Internet Path Changes. *ACM/IEEE Transactions on Networking*, 22(4):1025–1038, 2014.

[29] Italo Cunha, Pietro Marchetta, Matt Calder, Yi-Ching Chiu, Bruno VA Machado, Antonio Pescapè, Vasileios Giotsas, Harsha V Madhyastha, and Ethan Katz-Bassett. Sibyl: A Practical Internet Route Oracle. In *Proc. USENIX NSDI*, 2016.

[30] Wouter de Vries, José Jair Santanna, Anna Sperotto, and Aiko Pras. How Asymmetric is the Internet? In *Proc. IFIP AIMS*, 2015.

[31] Ramakrishnan Durairajan, Sathiya Kumaran Mani, Paul Barford, Robert Nowak, and Joel Sommers. TimeWeaver: Opportunistic One Way Delay Measurement Via NTP. In *Proc. IEEE ITC*, 2018.

[32] Brian Eriksson, Paul Barford, and Robert Nowak. Network Discovery from Passive Measurements. In *Proc. ACM SIGCOMM*, 2008.

[33] Xun Fan and John Heidemann. Selecting Representative IP Addresses for Internet Topology Studies. In *Proc. ACM IMC*, 2010.

[34] Tobias Flach, Ethan Katz-Bassett, and Ramesh Govindan. Quantifying Violations of Destination-based Forwarding on the Internet. In *Proc. ACM IMC*, 2012.

[35] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *Proc. USENIX NSDI*, 2015.

[36] Osvaldo Fonseca, Ítalo Cunha, Elverton Fazzion, Wagner Meira, Ronaldo A Ferreira, Ethan Katz-Bassett, et al. Identifying Networks Vulnerable to IP Spoofing. *IEEE Transactions on Network and Service Management*, 2021.

[37] Rodrigo Fonseca, George Porter, Randy Katz, Scott Shenker, and Ion Stoica. IP Options Are Not an Option. Technical report, EECS Department, University of California, Berkeley, 2005.

[38] Vasileios Giotsas, Thomas Koch, Elverton Fazzion, Ítalo Cunha, Matt Calder, Harsha V Madhyastha, and Ethan Katz-Bassett. Reduce, Reuse, Recycle: Repurposing Existing Measurements to Identify Stale Traceroutes. In *Proc. ACM IMC*, 2020.

[39] Brian J Goodchild, Yi-Ching Chiu, Rob Hansen, Haonan Lua, Matt Calder, Matthew Luckie, Wyatt Lloyd, David Choffnes, and Ethan Katz-Bassett. The Record Route Option is an Option! In *Proc. ACM IMC*, 2017.

[40] Matthieu Gouel, Kevin Vermeulen, Maxime Mouchet, Justin P Rohrer, Olivier Fourmaux, and Timur Friedman. Zeph & Iris Map the Internet: A Resilient Reinforcement Learning Approach to Distributed IP Route Tracing. *ACM SIGCOMM Computer Communication Review*, 52(1):2–9, 2022.

[41] Caitlin Gray, Clemens Mosig, Randy Bush, Cristel Pelsser, Matthew Roughan, Thomas C Schmidt, and Matthias Wahlisch. BGP Beacons, Network Tomography, and Bayesian Computation to Locate Route Flap Damping. In *Proc. ACM IMC*, 2020.

[42] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based Geolocation of Internet Hosts. *IEEE/ACM Transactions on Networking*, 14(6): 1219–1232, 2006.

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

[44] Gonca Gursun. Routing-Aware Partitioning of the Internet Address Space for Server Ranking in CDNs. *Computer Communications*, 106(C):86–99, July 2017.

[45] Gonca Gürsun, Natali Ruchansky, Evimaria Terzi, and Mark Crovella. Routing State Distance: A Path-based Metric for Network Analysis. In *Proc. ACM IMC*, 2012.

[46] Mohammad Javad Hajikhani, Thomas Kunz, and Howard Schwartz. A Recursive Method for Clock Synchronization in Asymmetric Packet-Based Networks. *IEEE/ACM Transactions on Networking*, 24(4):2332–2342, 2016.

[47] Yihua He, Michalis Faloutsos, Srikanth Krishnamurthy, and Bradley Huffaker. On Routing Asymmetry in the Internet. In *Proc. IEEE GLOBECOM*, 2005.

[48] Yuchen Huang, Michael Rabinovich, and Rami Al-Dalky. FlashRoute: Efficient Traceroute on a Massive Scale. In *Proc. ACM IMC*, 2020.

[49] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas E Anderson, and Arvind Krishnamurthy. PoiRoot: Investigating the Root Cause of Interdomain Path Changes. In *Proc. ACM SIGCOMM*, 2013.

[50] Ethan Katz-Bassett, John P John, Arvind Krishnamurthy, David Wetherall, Thomas E Anderson, and Yatin Chawathe. Towards IP Geolocation Using Delay and Topology Measurements. In *Proc. ACM IMC*, 2006.

[51] Ethan Katz-Bassett, Harsha V Madhyastha, John P John, Arvind Krishnamurthy, David Wetherall, and Thomas E Anderson. Studying Black Holes in the Internet with Hubble. In *Proc. USENIX NSDI*, 2008.

[52] Ethan Katz-Bassett, Harsha V Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter Van Wesep, Thomas E Anderson, and Arvind Krishnamurthy. Reverse Traceroute. In *Proc. USENIX NSDI*, 2010.

[53] Ethan Katz-Bassett, Colin Scott, David R Choffnes, Italo Cunha, Vytautas Valancius, Nick Feamster, Harsha V Madhyastha, Thomas E Anderson, and Arvind Krishnamurthy. LIFEGUARD: Practical Repair of Persistent Route Failures. In *Proc. ACM SIGCOMM*, 2012.

[54] Ken Keys. Internet-scale IP Alias Resolution Techniques. *ACM SIGCOMM Computer Communication Review*, 40(1):50–55, January 2010.

[55] Ken Keys, Young Hyun, Matthew Luckie, and Kim Claffy. Internet-scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM Transactions on Networking*, 21(2): 383–399, 2012.

[56] Rupa Krishnan, Harsha V Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas E Anderson, and Jie Gao. Moving beyond End-to-end Path Information to Optimize CDN Performance. In *Proc. ACM IMC*, 2009.

[57] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee. Internet Anycast: Performance, Problems and Potential. In *Proc. ACM SIGCOMM*, 2018.

[58] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and KC Claffy. AS Relationships, Customer Cones, and Validation. In *Proc. ACM IMC*, 2013.

[59] Harsha V Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas E Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proc. USENIX OSDI*, 2006.

[60] Harsha V Madhyastha, Ethan Katz-Bassett, Thomas E Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *Proc. USENIX NSDI*, 2009.

[61] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas E Anderson. User-Level Internet Path Diagnosis. In *Proc. ACM SOSP*, 2003.

[62] Ratul Mahajan, Ming Zhang, Lindsey Poole, and Vivek S Pai. Uncovering Performance Differences Among Backbone ISPs with Netdiff. In *Proc. USENIX NSDI*, 2008.

[63] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, Jonathan M Smith, et al. Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale. In *Proc. ACM IMC*, 2018.

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

[65] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. A Measurement Study of Internet Delay Asymmetry. In *Proc. Passive and Active Measurement Conference*, 2008.

[66] Andreas Reuter, Randy Bush, Italo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wahlisch. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *ACM SIGCOMM Computer Communication Review*, 48(1):19–27, Jan. 2018.

[67] Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. PEERING: Virtualizing BGP at the Edge for Research. In *Proc. ACM CoNEXT*, 2019.

[68] Justine Sherry, Ethan Katz-Bassett, Mary Pimenova, Harsha V Madhyastha, Thomas E Anderson, and Arvind Krishnamurthy. Resolving IP Aliases with Prespecified Timestamps. In *Proc. ACM IMC*, 2010.

[69] Rob Sherwood and Neil Spring. Touring the Internet in a TCP Sidecar. In *Proc. ACM IMC*, 2006.

[70] Rob Sherwood, Adam Bender, and Neil Spring. DisCarte: A Disjunctive Internet Cartographer. In *Proc. ACM SIGCOMM*, 2008.

[71] Jared M Smith, Kyle Birkeland, Tyler McDaniel, and Max Schuchard. Withdrawing the BGP Re-Routing Curtain: Understanding the Security Impact of BGP Poisoning through Real-World Measurements. In *Proc. USENIX NDSS*, 2020.

[72] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, 2002.

[73] Richard A Steenbergen. A Practical Guide to (Correctly) Troubleshooting with Traceroute. NANOG, 2009.

[74] Peng Sun, Laurent Vanbever, and Jennifer Rexford. Scalable Programmable Inbound Traffic Engineering. In *Proc. ACM SOSR*, 2015.

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

[76] Kevin Vermeulen, Justin P Rohrer, Robert Beverly, Olivier Fourmaux, and Timur Friedman. Diamond-Miner: Comprehensive Discovery of the Internet's Topology

Diamonds. In *Proc. USENIX NSDI*, 2020.
[77] Florian Wohlfart, Nikolaos Chatzis, Caglar Dabanoglu, Georg Carle, and Walter Willinger. Leveraging Interconnections for Performance: The Serving Infrastructure of a Large CDN. In *Proc. ACM SIGCOMM*, 2018.
[78] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a Lightweight Network Location Service without Virtual Coordinates. In *Proc. ACM SIGCOMM*, 2005.
[79] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *Proc. USENIX NSDI*, 2007.
[80] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Tariq Mukarram, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proc. ACM SIGCOMM*, 2017.
[81] Ming Zhang, Chi Zhang, Vivek S Pai, Larry L Peterson, and Randolph Y Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-area Services. In *Proc. USENIX OSDI*, 2004.
[82] Xiao Zhang, Tanmoy Sen, Zheyuan Zhang, Tim April, Balakrishnan Chandrasekaran, David Choffnes, Bruce M Maggs, Haiying Shen, Ramesh K Sitaraman, and Xiaowei Yang. AnyOpt: Predicting and Optimizing IP Anycast Performance. In *Proc. ACM SIGCOMM*, 2021.
[83] Zheng Zhang, Ying Zhang, Y. Charlie Hu, Z. Morley Mao, and Randy Bush. iSPY: Detecting IP Prefix Hijacking on My Own. In *Proc. ACM SIGCOMM*, 2008.
[84] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network Neutrality Inference. In *Proc. ACM SIGCOMM*, 2014.
[85] Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Matt Calder, and Ethan Katz-Bassett. The Best of Both Worlds: High Availability CDN Routing Without Compromising Control. In *Proc. ACM IMC*, 2022.