Locating and Enumerating Anycast: a Comparison of Two Approaches

Remi Hendriks
University of Twente
Enschede, the Netherlands
remi.hendriks@utwente.nl

Tim Betzer
Technical University of Munich
Munich, Germany
betzer@net.in.tum.de

Ben Du
CAIDA/UC San Diego
La Jolla, CA, USA
bendu@ucsd.edu

Raffaele Sommese
University of Twente
Enschede, the Netherlands
r.sommese@utwente.nl

Mattijs Jonker
University of Twente
Enschede, the Netherlands
m.jonker@utwente.nl

Roland van Rijswijk-Deij
University of Twente
Enschede, the Netherlands
r.m.vanrijswijk@utwente.nl

Abstract

Anycast allows for providing services from multiple, geographically distant Points of Presence (PoPs), using a single IP address, to, *e.g.*, improve resilience. Due to its opaqueness, it is often unknown which addresses are provisioned using anycast and, if so, where the PoPs are located. As anycast is widely used for critical Internet infrastructures (*e.g.*, the DNS) efforts have been made to map anycast deployments. The current state-of-the-art mapping technique, iGreedy, relies on latency-based measurements, and is adversely affected by noise caused by, *e.g.*, network processing delays. Previous work has shown that traceroute can alternatively be used to detect anycast. As traceroute reveals the hops a packet traverses, it may also be used to locate sites using geolocation data for hops near the anycast PoPs.

This paper is the first to assess the performance of the traceroute-based approach at scale, by targeting 14k prefixes from an anycast census. Using ground truth we show traceroute achieves a slight increase in enumeration and geolocation precision over iGreedy. However, it suffers from overestimating the number of PoPs and incurs a 4× increase in probing cost, making it unattractive for anycast censuses.

CCS Concepts

• **Networks** → **Network measurement**.

Keywords

Anycast; Geolocation; Traceroute

ANRW '25, Madrid, Spain

ACM ISBN 979-8-4007-2009-3/2025/07
https://doi.org/10.1145/3744200.3744783

ACM Reference Format:
Remi Hendriks, Tim Betzer, Ben Du, Raffaele Sommese, Mattijs Jonker, and Roland van Rijswijk-Deij. 2025. Locating and Enumerating Anycast: a Comparison of Two Approaches. In *Applied Networking Research Workshop (ANRW '25), July 22, 2025, Madrid, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3744200.3744783

1 Introduction

Anycast is the practice of making an IP address available in multiple discrete locations [20]. This allows operators to offer replicated services nearer to clients and increase resilience through redundancy. Examples include DNS resolvers and nameservers, and CDNs providing web caching. While the number of anycast addresses on the Internet is small (<0.1% [18]) they serve a large share of Internet traffic [6].

For this reason, efforts have been made to map anycast deployments on the Internet to *e.g.*, measure Internet resilience. The most notable are iGreedy [7] and MAnycast^2 [27], where the former, iGreedy, also enumerates and geolocates PoPs behind anycast addresses. iGreedy geolocates using latency measurements by probing an anycast address from multiple geographically distributed Vantage Points (VPs), similar to conventional IP unicast geolocation [32]. However, such latency measurements are known to be noisy due to *e.g.*, network processing and queuing delays.

Unicast IP geolocation often involves the use of tools such as traceroute to improve precision [32]. Traceroute allows for measuring the path that an Internet packet takes to reach its target. When determining the location of a target, in case no location information is available for the target itself, traceroute can be used to find nearby hops for which location information is available. It has also been used to geolocate anycast using the location of hops near anycast PoPs, to infer the location of PoPs themselves [31]. One example is verifying GDPR compliance when sending personal data to anycast addresses [24]. However, traceroute has limitations: operators may block the ICMP Time Exceeded messages

traceroute relies on, and networks that deploy Multiprotocol Label Switching (MPLS) tunneling may hide traceroute hops [12], making it infeasible to geolocate them.

This work investigates the use of traceroute for anycast censuses by performing a large-scale measurement toward 14k anycast prefixes from an anycast census [18]. We show traceroute finds an average of 11.82% more PoPs, yet may also output multiple neighboring locations for a single PoP resulting in overestimation of the number of anycast sites. Using ground truth we show traceroute achieves a mean geolocation error of 26km compared to 51km using iGreedy. However, due to a 4× increase in probing cost, we argue traceroute-based anycast censuses are nonpractical. We make all code and data publicly available[1].

This paper is structured as follows. First, in §2 we discuss background on anycast detection and traceroute followed with related work on using traceroute to detect anycast. Then, we detail the methodology used in §3 and provide our results in §4. Finally, we discuss the performance of traceroute to detect anycast and list further use cases in §5.

2 Background and related work

We present background on traceroute and anycast detection, and discuss work on measuring anycast using traceroute.

2.1 Traceroute

Traceroute traces Internet paths by triggering routers on the path towards a target to send back ICMP `Time-Exceeded` replies. Such routers are often configured with DNS `PTR` records that operators use for debugging purposes, and may contain information such as the ASN, network type, country and city where the router is located. IP to location databases may also have known locations for traceroute hops. This makes it possible to infer (part of) the geographical path of traceroutes. Furthermore, it is possible to infer locations of Internet addresses using geolocation information available from nearby hops (*i.e.*, hops with similar RTT values). In particular, the pen-ultimate hop (*p-hop*), *i.e.*, the hop before the destination, is often used to geolocate IP addresses.

Several works use traceroute to perform unicast geolocation [32], in this work we use it to geolocate anycast PoPs.

2.2 Anycast detection

The current state-of-the-art technique to detect anycast is iGreedy [7]. By measuring the latency from multiple VPs to a target, it finds Great-Circle Distances (GCD) using the speed of packets in fibre optic cables (roughly two thirds the speed of light). In the case of a unicast target, all circles will overlap providing a single solution that resides in the intersection. However, in the case of anycast it finds non-intersecting sets of circles, as VPs reach different anycast PoPs. In this case, the iGreedy algorithm finds the minimum set of independent overlapping areas in which PoPs must be located for there to be no speed-of-light violation. Next, it geolocates anycast PoPs by selecting an airport near the overlapping area using a metric that includes the population of the main city that the airport serves, since operators often deploy PoPs in large metropolitan areas to maximize utility.

iGreedy was shown to be quite accurate, achieving a recall of over 50% with an average geolocation error of 361km. In the end, the accuracy of iGreedy depends on the number of VPs used and their geographical/topological diversity.

Similarly, RIPE IPmap, a geolocation platform operated by the RIPE NCC, detects anycast using its active geolocation engine. It classifies an IP address as anycast if multiple globally distributed vantage points observe the target IP address at a latency of 1 ms or less [13, 23]. Note that this likely leads to a gross underestimation of the number of anycast IPs and sites, since 1 ms is a very tight bound.

Another detection technique for anycast, MAnycast[2], is achieved by measuring anycast using anycast[27]. This lightweight technique does not allow for geolocating anycast.

2.3 Measuring anycast with traceroute

Xun et al. analysed the usage of anycast in the DNS in 2013 [14] by leveraging CHAOS records, which allow for the specific nameserver reached to be identified [30]. As unique CHAOS records were used for multiple load-balanced nameservers at a single anycast PoP, the authors augmented their approach with traceroute to resolve ambiguities.

Next, in 2017, Wei et al. measured the occurrence of anycast flipping (*i.e.*, a single client flipping between multiple anycast PoPs) and used the pen-ultimate hops of traceroutes to determine the reached PoP [29].

More recently, in 2023, Zhou et al. used traceroute to geolocate the PoPs of regional anycast deployments [31] By geolocating the p-hop as observed from RIPE Atlas VPs, they infer the location of the PoP reached by mapping it to the closest PoP location from available ground truth. They performed their methodology towards regional anycast prefixes from two CDNs and showed they were able to infer the reached site in the majority of cases.

Pascual et al. introduced a tool to trace anycast communications (*Hunter*) in 2024, to verify compliance to data protection regulations for personal data transfers [24]. Their method geolocates the p-hop visible in a traceroute, then performs geolocation using latency measurements from nearby RIPE Atlas VPs. Their method showed high accuracy and precision in geolocating the PoP reached, but was only validated using two anycast addresses from a single operator.

Unlike previous works, which used traceroute towards a select few anycast deployments, we explore the effectiveness

[1] https://github.com/ut-dacs/anycast-trace-locator

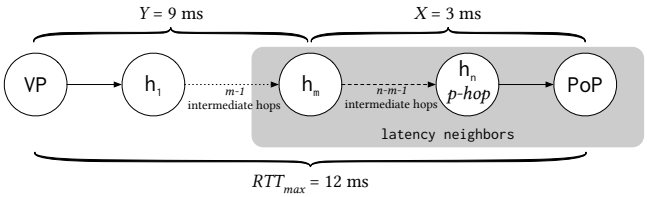

Figure 1: Traceroute latency parameters as per [10]

of using traceroute to enumerate and geolocate anycast PoPs using 14k anycast prefixes from a public anycast census [18].

3 Methodology

In this section we describe our methodology that builds on previous work. Our main contribution is a scalable implementation that does not rely on *e.g.*, CHAOS records, or ground-truth to cross-reference the PoP reached.

3.1 Latency propagation

Latency propagation is a technique used in traditional unicast geolocation, where traceroute hops with similar round-trip-times (RTTs) are classified as so-called latency neighbors and inferred to be near each other. Dan et al. assessed the accuracy in geolocating using latency propagation [10]. Their methodology uses a latency threshold X which determines the maximum RTT difference between two hops for them to be considered latency neighbors. A second parameter Y determines the maximum RTT difference between the VP and the closest latency neighbor. Their analysis showed that using values of 3 ms for X and 9 ms for Y incurs a median error of 10.1 km and outperforms IP to location databases. Using lower parameter values reduces the median error, at a cost in coverage (*i.e.*, for a PoP to be located it requires a VP with less than $X + Y$ RTT). We use the same parameters in this work, meaning there must be a latency neighbor within 9 ms of the VP, with at most 3 ms RTT difference from the PoP. This requires the VP to be within 12 ms of the PoP. Fig. 1 schematically shows these constraints.

Previous work [14, 24, 31] uses the penultimate hop *(p-hop)* to infer the location of the PoP. We later show this approach yields false locations as the *p-hop* may be distant from the PoP (*e.g.*, when there is tunneling). Furthermore, the *p-hop* may have no geolocation information available (*e.g.*, it responds with a bogon address) whereas our approach may use other hops near the PoP (within the latency threshold) that have location data.

3.2 Locating traceroute hops

To infer the location of the PoP, we require the location of its latency neighbors. When the VP itself is a latency neighbor (*i.e.*, within 3ms of the target), we infer the target to be near the VP (which has a known location). Otherwise, we attempt to geolocate traceroute hops using PTR records. This is achieved with Hoiho [22] that extracts geographical information from PTR records using regular expressions obtained using ITDK [4]. However, PTR records are not always available, may not contain geo-hints, or have an unknown regex pattern in which case we use IPInfo [19], an IP to location database. We select IPInfo as it shown to be more accurate than other commercial datasets [11]. To minimize the impact of wrong geo-hints in PTR records and wrong locations from IP to location databases, we validate hop locations using the measured latency from the VP similar to Zhou et al. [31]

3.3 Inferring PoP location

Using the latency neighbor technique we may find multiple cities that neighbor a single PoP. For this reason, we reuse iGreedy's airport selection algorithm [7] that uses the distance from the inferred location and the population of the nearby metropolitan area that the airport serves as heuristics. We attempt to find an airport within 100 km from the latency neighbor, if it exists, else we take the nearest airport.

3.4 Limitations

Hidden hops. Tunneling and routers configured to not send ICMP Time Exceeded replies will result in hidden traceroute hops that may lead to indeterminate results.

Hop latency estimates. The RTTs towards traceroute hops are estimates of the actual latency between the VP and the hop itself. Due to inflated RTTs (caused by *e.g.*, network processing delays), one may infer a hop to be artificially close to the target. This could lead to false classifications of latency neighbors. To combat this, we only classify latency neighbors as valid if they are within 9 ms of the VP as such hops are geographically close to the VP and generally suffer less from network processing delays.

Grouping locations. When performing a traceroute from multiple VPs to a single target, *e.g.*, a single PoP, we will find multiple neighboring routers surrounding the target. These routers may be at most 6 ms distant from each other, as a latency neighbor is within 3 ms of the target, this roughly translates to a possible distance of 600 km. To avoid counting multiple PoPs in such situations, we group results using iGreedy's algorithm that locates the most likely nearby airport based on its distance and population it serves. However, we may find multiple major airports surrounding a single PoP where our method overestimates the number of PoPs.

To avoid the possibility of overcounting, in the worst case, requires to group locations found within 600 km of each other as a single PoP. Yet, by doing so it severely reduces performance as PoPs within such radius are no longer distinguishable (*e.g.*, the distance between Amsterdam and Berlin is less than 600 km). We later show that using a radius of

100 km provides the best trade-off between performance and overestimating PoPs using ground truth.

3.5 Measurement setup

We use all VPs of CAIDA's *Archipelago* (Ark), covering 66 countries, from which we perform traceroute measurements. We choose this platform as it provides accurate locations of VPs, has good global coverage, and allows for large scale traceroute measurements [21]. Furthermore, Ark implements Paris traceroute [2] which avoids anomalous traceroute results due to multi-path topologies caused by load-balancers. Using Ark we send a single probe per traceroute hop, unlike traditional traceroute that sends 3 probes per hop, to limit the probing burden. This analysis can be repeated using RIPE Atlas, and other probing platforms, though it would require filtering of Atlas probes with wrong user reported locations.

We recall that our goal is to analyse how well traceroute can enumerate and geolocate PoPs of an anycast deployment. In order to reduce the impact of our experiments, we therefore chose to target a list of 14k known anycast prefixes that were confirmed using iGreedy [18]. Furthermore, we compare against our own iGreedy measurement using GCD from ping latencies, which we run using the same VPs offered by Ark. Finally, we make use of publicly available information of root servers to determine the precision of both methodologies in geolocating PoPs [26].

4 Results

In total, our dataset consists of traceroutes measured between 274 VPs (inside 178 distinct ASes) and 13,765 target anycast /24-prefixes. For 30 prefixes no completed traceroutes were captured, *e.g.*, because the AS or one of its upstreams blocks ICMP Time Exceeded messages, a known limitation of traceroute. Manual inspection shows they are ping responsive but unreachable with traceroute. The remaining 13,735 targets (inside 906 distinct ASes) consist of 3.35 million traceroutes, averaging 244 completed traceroutes per target. The total number of traceroute hops captured is 27.2 million, averaging 8.12 hops captured for each *(VP, target)* pair.

In total, our traceroutes traverse 1,827 distinct ASes and we observe 99,939 unique hop addresses (including 6,282 bogon addresses). Using OpenINTEL [28] we obtain PTR records for 45,106 addresses of which 19,613 (43.5%) were translated using Hoiho [22] and 16,565 (36.7%) contained city-level geolocation. Next, using IPInfo we obtain city-level geolocation for all non-bogon addresses. We validate locations by assessing whether it is within possible distance of the VP using the measured RTT towards that hop. Whenever a valid location is available we use it, preferring PTR record locations over IPInfo. In total, we have 16,929 (16.9%) hops with a valid PTR location and 75,143 (75.2%) with a valid IPInfo location. For 7,867 (7.9%) hops there is no valid location available.

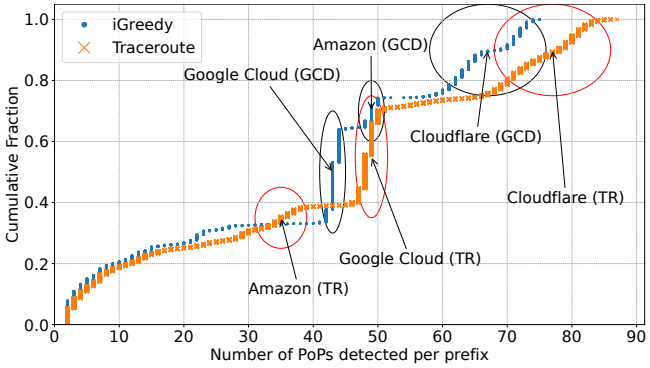

Figure 2: Distribution of PoPs found using the iGreedy and traceroute technique for 13,765 anycast prefixes.

In total, we confirm 13,478 prefixes to be anycast using the latency neighbor methodology. For the 257 missing prefixes, 197 are observed to have two PoPs using GCD and 29 to have three PoPs. These are mostly easy to detect cases of anycast using GCD, as the PoPs are geographically distant from each other. However, they only have an Ark VP within 12 ms for at most one PoP (*i.e.*, the remaining PoPs are not visible with the latency neighbor method). The remaining 31 prefixes failed due to a low number of VPs providing completed traceroutes.

4.1 Enumeration

Fig. 2 shows the CDF for the number of PoPs detected by both methodologies. We find that for small to medium sized anycast deployments (< 30 PoPs, bottom left) both methodologies detect a similar number of sites. Next, for large deployments (> 40 PoPs, top right) we observe that traceroute consistently finds more PoPs. Overall, the traceroute technique finds 11.82% more PoPs compared to iGreedy.

Most of the large deployments can be attributed to a few CDNs. Since the additional benefit of traceroute in terms of enumeration is most visible for large deployments, Table 1 provides a break-down on the enumeration counts for the 5 largest ASes in our dataset alongside ground truth. For ground truth we use public information on the number of cities where PoPs are deployed. Operators may deploy prefixes using different configurations including different numbers of PoPs used. For example, Fastly has PoPs in 79 distinct cities [16] but offers *intelligent PoP placement* [15] where selective deployments are offered to customers.

We find the traceroute technique achieves a higher enumeration count for all CDNs except Amazon. Investigating these targets, we observe no traceroute hops are visible inside Amazon's network. As our methodology requires a latency neighbor within 3 ms of the target, this means that any routing longer than 3 ms inside Amazon's network results in an undetected PoP.

AS	Organization	IPv4 (/24)	TR	GCD	GT
396982	Google Cloud	3,915	48.4	43.2	103 [17]
13335	Cloudflare	3,096	74.2	65.9	335 [9]
16509	Amazon	1,325	40.2	48.6	>300 [3]
54113	Fastly	799	16.5	13.4	79 [16]
209242	Cloudflare Spectrum	306	74.6	63.7	335 [9]

Table 1: Largest ASes originating anycast with the mean PoP count found using traceroute (TR), iGreedy (GCD), and the PoP count from public data (GT).

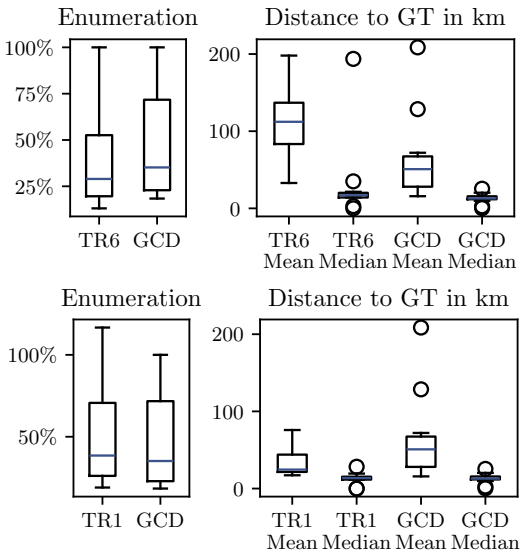

Figure 3: Comparison between traceroute with 600 km (TR6) and 100 km (TR1) radius and iGreedy (GCD) results using the root server locations as ground truth (GT). Enumeration compares identified locations to GT; values over 100% indicate overcounting.

4.2 Geolocation

Fig. 3 compares traceroute to iGreedy using all DNS root servers as ground truth except the G-root server since it is unresponsive to pings. The left-side of the figure shows a boxplot for enumeration counts, and we show the mean and median errors for geolocation on the right. As discussed previously we compare results when grouping using a 600 km radius (top) and a 100 km radius (bottom)

For iGreedy we find a recall of 35.19% PoPs detected and no overcounting. The median for mean distances is 50.81 km and for median distances it is 12.94 km. Traceroute with a grouping radius of 600 km has a median enumeration of 28.97% with no overcounting. The median for mean distances is more than double compared to iGreedy with 112.18 km and for median distances amounts to 17.02 km. Next, a grouping radius of 100 km leads to overcounting in two instances with a median enumeration of 38.53%, while reducing the median for mean distances to 25.58 km and to 13.57 km for median distances. Overall, we find that the grouping radius provides

Letter	GT	GCD Count	TR Count	GCD Geo error	TR Geo error
A	20	0.7000	0.7500	15.7548	42.7639
B	6	1.0000	1.1667	22.6511	54.0090
C	13	0.7692	0.6923	59.6628	47.5731
D	222	0.2748	0.3108	18.2553	22.7183
E	325	0.2308	0.2615	29.8686	21.0821
F	359	0.1838	0.2089	61.9196	21.7074
H	12	1.0000	1.0833	208.6422	75.8460
G	6	0.0000	0.0000	-	-
I	85	0.3176	0.3647	72.0640	29.2808
J	101	0.3861	0.4059	41.9601	25.2775
K	121	0.2231	0.1901	128.5593	19.2221
L	123	0.2033	0.2602	34.7577	23.8902
M	23	0.6087	0.5217	65.7767	17.2038

Table 2: Breakdown of results per Root Server (Letter) with number of PoPs (GT), ratio of PoPs found (GCD, TR Count) and mean geographical error (GCD, TR Geo). Red indicates overcounting.

a trade-off where a lower radius yields more accurate geolocation results, but increases the occurrence of overcounting. Despite the overcounting, we use a 100 km grouping factor as it yields more accurate results. We provide a breakdown of results per root-letter in Table 2.

4.3 Probing cost

A large limitation of traceroute is probing cost as traceroute requires multiple packets to measure a path. In our dataset we observe an average of 9.83 hops including the destination. Compared to iGreedy, which measures the latency with a single ping packet this is nearly a 10-fold increase in probing cost. However, we can reduce the probing cost by skipping initial hops adjacent to the VP and inside the same AS. On average, we find an average of 2.87 of same-AS hops lowering the mean count to 6.96 hops. This is a maximum on the number of hops that may be skipped, as VPs may have an inconsistent number of same-AS hops.

Next, we can reduce the probing cost even further as we require a measured RTT below 12 ms between the VP and target (to satisfy the latency neighbor constraints). Therefore, using an initial ping measurement from all VPs we can limit traceroutes to targets within this threshold. On average we find 124 out of 274 VPs are within 12 ms of the anycast target. This constraint also reduces the average hop count decreases from 9.83 to 8.40, as these VPs are closer to the target, which can be reduced to 5.53 when skipping same-AS hops.

We provide a breakdown of the probing cost in Table 3. This shows the probing cost can be reduced to 795 probes per target, by performing a single latency measurement from all 274 probes, followed with a traceroute measurement from an average of 124 VPs within 12 ms that measure an average of 4.20 hops when skipping same-AS hops. Unsurprisingly,

Technique	Probes	# of VPs	Total
TR (full)	9.83	274	2,693
TR (skipping same-AS hops)	6.96	274	1,907
TR (latency constrained)	8.40*	124†	1,316
TR (reduced)	5.53*	124†	960
iGreedy	1	274	274

*Initial measurement required from all 274 VPs.
†Average number of VPs within 12 ms RTT.

Table 3: Probing cost per target breakdown.

we find that VPs with a low latency traverse a lower number of hops to reach the target.

4.4 Using p-hops

As mentioned, previous work used the *p-hop* to assess the location of the anycast site reached. However, we only use *latency neighbors* that are within 3 ms of the target for VPs with at most 12 ms RTT towards the target. To assess the accuracy of relaxing these latency constraints to use all *p-hops* measured, we perform an unconstrained measurement towards the DNS root letters. On average, this finds 11.92 more sites for each root letter. Yet it also amplifies the overestimation error motivating our decision to use the latency neighbor constraints. For example, B-root has 6 PoPs where the unconstrained *p-hop* method finds 13 PoPs and for H-root with 12 PoPs it finds 30.

However, similar to Zhou et al. we can cross-reference the PoP reached using *p-hop* locations [31]. In total, we have 3.35 million *p-hops* (averaging 243 per target) in our dataset, 1.88 million of which are invalid latency neighbors. We cross-reference these invalid hops with inferred locations using valid latency neighbors. Figure 4 shows the average distance between invalid *p-hops* and inferred locations. Overall, we find that the majority of *p-hops* can be correlated to a PoP found within a 40 km radius, therefore this method is valid when cross-referencing locations with *e.g.*, ground-truth or iGreedy locations. However, a long tail with large distances cannot be correlated to a nearby PoP location that may lead to overcounting when used to infer additional locations.

5 Discussion

Our findings show that traceroute outperforms iGreedy in terms of enumeration and geolocation. More specifically, we find that traceroute achieves an average increase of 11.82% in enumerating anycast PoPs and has a lower mean geolocation error (26 km compared to 51 km). However, due to, *e.g.*, unreliable hop latencies it may overestimate the number of PoPs and it comes at a significant increase in probing cost compared to iGreedy. Therefore, we do not recommend to use traceroute as a method to perform anycast censuses.

Our results do indicate other benefits to performing traceroutes to anycast, such as mapping which ASes reach a particular PoP and mapping topological properties of anycast deployments (*e.g.*, the upstream ASes for a particular PoP).

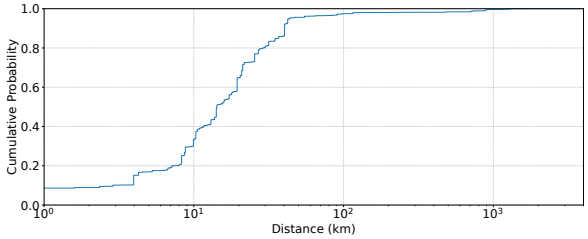

Figure 4: Distances (log scale) between invalid latency neighbor *p-hops* and the nearest airport found using the latency neighbor method.

5.1 Future work

Longitudinal analysis. Like IPMap [23], our methodology can be performed using historical traceroute data. RIPE Atlas, Measurement Lab (M-Lab), and Ark have public longitudinal datasets of traceroute measurements. Most notable is FANTAIL [1] which provides access to historical traceroute data towards all routable /24 IPv4 prefixes [8]. Our analysis can be repeated using such traceroute data to map the development of anycast infrastructures over time, giving insights in, *e.g.*, infrastructure expansions and topological changes.

Mapping anycast topologies. We can infer the ASes traversed in traceroutes and identify upstream ASes for particular PoPs using CAIDA's prefix2as dataset [5]. Using this method we can find a lower-bound on the upstream ASes present at each anycast PoP. With *e.g.*, RIPE Atlas, this analysis can be performed from a large number of origin ASes to obtain a more complete picture.

Sub-optimal anycast routing. As done by Rizvi et al. [25] traceroute can be used to detect causes of sub-optimal anycast routing (*i.e.*, a client routing to a distant anycast site when a nearby one is available). Our methodology can extend their work as operators can use it to infer the PoP receiving traffic from distant clients. Furthermore, whilst Rizvi et al. looked at the penultimate AS, we observe adjacent hops in distinct ASes with large latency increases on-path to the anycast AS. We suspect these are remote-peering links.

Further refinement. The average distance between p-hops (that are invalid latency neighbors) and the inferred anycast location suggest that the latency neighbor constraints may be too restrictive. Furthermore, false latency neighbors can possibly be avoided by performing multiple traceroutes and taking the minimum of observed latencies to each hop. However, this would come at a significant increase in probing cost. Finally, the grouping radius may be altered to improve geolocation accuracy at the cost of overcounting, or vice versa. We leave these parameters adjustable in our code[2].

[2]https://github.com/ut-dacs/anycast-trace-locator

Acknowledgments

The CAIDA Ark platform is supported by U.S. NSF grants OAC-2131987, CNS-2120399, and CNS-2212241. The views and conclusions are those of the authors and do not necessarily represent endorsements, either expressed or implied, of NSF. This work has been partially funded by CATRIN (NWO grant NWA.1215.18.003). This research was made possible by OpenINTEL, a joint project of the University of Twente, SURF, SIDN, and NLnet Labs.

References

[1] 2020. FANTAIL: Facilitating Advances in Network Topology Analysis. https://catalog.caida.org/presentation/2020_fantail_kismet. https://doi.org/presentation/2020_fantail_kismet Accessed: 29-04-2025.

[2] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. 2006. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of the 6th ACM IMC*. 153–158.

[3] Amazon (AWS). 2025. Global Edge Network. https://aws.amazon.com/cloudfront/features. [Accessed 25-04-2025].

[4] CAIDA. 2024. Macroscopic Internet Topology Data Kit — caida.org. https://www.caida.org/catalog/datasets/internet-topology-data-kit/release-2024-08/. [Accessed 13-06-2025].

[5] CAIDA. 2024. Routeviews Prefix to AS mappings Dataset (pfx2as) for IPv4 and IPv6 — caida.org. https://caida.org/catalog/datasets/routeviews-prefix2as. [Accessed 29-04-2025].

[6] Danilo Cicalese, Danilo Giordano, Alessandro Finamore, Marco Mellia, Maurizio Munafò, Dario Rossi, and Diana Joumblatt. 2021. A First Look at Anycast CDN Traffic. arXiv:1505.00946 [cs.NI] https://arxiv.org/abs/1505.00946

[7] Danilo Cicalese, Diana Joumblatt, Dario Rossi, Marc-Olivier Buob, Jordan Augé, and Timur Friedman. 2015. A fistful of pings: Accurate and lightweight anycast enumeration and geolocation. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2776–2784. https://doi.org/10.1109/INFOCOM.2015.7218670

[8] Kimberly Claffy, Young Hyun, Ken Keys, Marina Fomenkov, and Dmitri Krioukov. 2009. Internet Mapping: From Art to Science. In *2009 Cybersecurity Applications & Technology Conference for Homeland Security*. 205–211. https://doi.org/10.1109/CATCH.2009.38

[9] Cloudflare. 2025. Network edge locations. https://cloudflare.com/network. [Accessed 29-04-2025].

[10] Ovidiu Dan, Vaibhav Parikh, and Brian D. Davison. 2021. IP Geolocation Using Traceroute Location Propagation and IP Range Location Interpolation. In *Companion Proceedings of the Web Conference (WWW '21)*. Association for Computing Machinery, 332–338. https://doi.org/10.1145/3442442.3451888

[11] Omar Darwich, Hugo Rimlinger, Milo Dreyfus, Matthieu Gouel, and Kevin Vermeulen. 2023. Replication: Towards a Publicly Available Internet Scale IP Geolocation Dataset. In *Proceedings of the 23rd ACM IMC (IMC '23)*. Association for Computing Machinery, 1–15. https://doi.org/10.1145/3618257.3624801

[12] Benoit Donnet, Matthew Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. 2012. Revealing MPLS tunnels obscured from traceroute. *SIGCOMM Comput. Commun. Rev.* 42, 2 (March 2012), 87–93. https://doi.org/10.1145/2185376.2185388

[13] Ben Du, Massimo Candela, Bradley Huffaker, Alex C. Snoeren, and kc claffy. 2020. RIPE IPmap active geolocation: mechanism and performance evaluation. *SIGCOMM Comput. Commun. Rev.* 50, 2 (May 2020), 3–10. https://doi.org/10.1145/3402413.3402415

[14] Xun Fan, John Heidemann, and Ramesh Govindan. 2013. Evaluating anycast in the domain name system. In *2013 Proceedings IEEE INFOCOM*. 1681–1689. https://doi.org/10.1109/INFCOM.2013.6566965

[15] Fastly. 2025. Managed CDN. https://www.fastly.com/services/managed-cdn. [Accessed 25-04-2025].

[16] Fastly. 2025. Network Map. https://www.fastly.com/network-map. [Accessed 25-04-2025].

[17] Google. 2025. Network edge locations. https://cloud.google.com/vpc/docs/edge-locations. [Accessed 25-04-2025].

[18] Remi Hendriks, Matthew Luckie, Mattijs Jonker, Raffaele Sommese, and Roland van Rijswijk-Deij. 2025. MAnycast Reloaded: a Tool for an Open, Fast, Responsible and Efficient Daily Anycast Census. arXiv:2503.20554 [cs.NI] https://arxiv.org/abs/2503.20554

[19] ipinfo.io. 2025. Trusted IP Data Provider, from IPv6 to IPv4. https://ipinfo.io. [Accessed 29-04-2025].

[20] Kurt Erik Lindqvist and Joe Abley. 2006. Operation of Anycast Services. RFC 4786. https://doi.org/10.17487/RFC4786

[21] Matthew Luckie, Shivani Hariprasad, Raffaele Sommese, Brendon Jones, Ken Keys, Ricky Mok, and K Claffy. 2025. An Integrated Active Measurement Programming Environment. In *International Conference on Passive and Active Network Measurement*. Springer, 137–152.

[22] Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and K Claffy. 2021. Learning to extract geographic information from internet router hostnames. In *Proceedings of the 17th International CoNEXT (CoNEXT '21)*. Association for Computing Machinery, 440–453. https://doi.org/10.1145/3485983.3494869

[23] RIPE NCC. 2025. IPmap. https://ipmap.ripe.net/. [Accessed 29-04-2025].

[24] Hugo Pascual, Jose M. del Alamo, David Rodriguez, and Juan C. Dueñas. 2024. Hunter: Tracing anycast communications to uncover cross-border personal data transfers. *Computers & Security* 141 (2024), 103823. https://doi.org/10.1016/j.cose.2024.103823

[25] A. S. M. Rizvi, Tingshan Huang, Rasit Esrefoglu, and John Heidemann. 2024. Anycast Polarization in the Wild. In *Passive and Active Measurement*, Philipp Richter, Vaibhav Bajpai, and Esteban Carisimo (Eds.). Springer Nature Switzerland, 104–131.

[26] root servers.org. 2025. Root server locations. https://root-servers.org/. [Accessed 25-04-2025].

[27] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2020. MAnycast2: Using Anycast to Measure Anycast. In *Proceedings of the 20th ACM IMC (IMC '20)*. Association for Computing Machinery, 456–463. https://doi.org/10.1145/3419394.3423646

[28] Roland van Rijswijk-Deij, Mattijs Jonker, Anna Sperotto, and Aiko Pras. 2016. A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. *IEEE JSAC* 34, 6 (2016), 1877–1888. https://doi.org/10.1109/JSAC.2016.2558918

[29] Lan Wei and John Heidemann. 2018. Does Anycast Hang Up on You (UDP and TCP)? *IEEE Transactions on Network and Service Management* 15, 2 (2018), 707–717. https://doi.org/10.1109/TNSM.2018.2804884

[30] S. Woolf and D. Conrad. 2007. *Requirements for a Mechanism Identifying a Name Server Instance*. RFC 4892. https://rfc-editor.org/rfc/rfc4892

[31] Minyuan Zhou, Xiao Zhang, Shuai Hao, Xiaowei Yang, Jiaqi Zheng, Guihai Chen, and Wanchun Dou. 2023. Regional IP Anycast: Deployments, Performance, and Potentials. In *Proceedings of the ACM SIGCOMM Conference (ACM SIGCOMM '23)*. Association for Computing Machinery, 917–931. https://doi.org/10.1145/3603269.3604846

[32] Aviram Zilberman, Adi Offer, Bar Pincu, Yoni Glickshtein, Roi Kant, Oleg Brodt, Andikan Otung, Rami Puzis, Asaf Shabtai, and Yuval Elovici. 2024. A Survey on Geolocation on the Internet. *IEEE Communications Surveys & Tutorials* (2024), 1–1. https://doi.org/10.1109/COMST.2024.3518398