Old but Gold: Prospecting TCP to Engineer and Live Monitor DNS Anycast

Giovane C. M. Moura[1,2](✉), John Heidemann[3], Wes Hardaker[3], Pithayuth Charnsethikul[3], Jeroen Bulten[4], João M. Ceron[1], and Cristian Hesselman[1,5]

[1] SIDN Labs, Arnhem, The Netherlands
giovane.moura@sidn.nl
[2] TU Delft, Delft, The Netherlands
[3] USC/ISI, Marina Del Rey, USA
[4] SIDN, Arnhem, The Netherlands
[5] University of Twente, Enschede, The Netherlands

Abstract. DNS latency is a concern for many service operators: CDNs exist to reduce service latency to end-users but must rely on global DNS for reachability and load-balancing. Today, DNS latency is monitored by active probing from distributed platforms like RIPE Atlas, with Verfploeter, or with commercial services. While Atlas coverage is wide, its 10k sites see only a fraction of the Internet. In this paper we show that passive observation of TCP handshakes can measure *live DNS latency, continuously, providing good coverage of current clients of the service*. Estimating RTT from TCP is an old idea, but its application to DNS has not previously been studied carefully. We show that there is sufficient TCP DNS traffic today to provide good operational coverage (particularly of IPv6), and very good temporal coverage (better than existing approaches), enabling near-real time evaluation of DNS latency from *real clients*. We also show that DNS servers can optionally solicit TCP to broaden coverage. We quantify coverage and show that estimates of DNS latency from TCP is consistent with UDP latency. Our approach finds previously unknown, real problems: *DNS polarization* is a new problem where a hypergiant sends global traffic to one anycast site rather than taking advantage of the global anycast deployment. Correcting polarization in Google DNS cut its latency from 100 ms to 10 ms; and from Microsoft Azure cut latency from 90 ms to 20 ms. We also show other instances of routing problems that add 100–200 ms latency. Finally, *real-time* use of our approach for a European country-level domain has helped detect and correct a BGP routing misconfiguration that detoured European traffic to Australia. We have integrated our approach into several open source tools: `ENTRADA`, our open source data warehouse for DNS, a monitoring tool (`Anteater`), which has been operational for the last 2 years on a country-level top-level domain, and a DNS anonymization tool in use at a root server since March 2021.

O. Hohlfeld et al. (Eds.): PAM 2022, LNCS 13210, pp. 264–292, 2022.
https://doi.org/10.1007/978-3-030-98785-5_12

1 Introduction

Latency is a key performance indicator for many DNS operators. DNS latency is seen as a bottleneck in web access [65]. Content Delivery Networks (CDNs) are particularly sensitive to DNS latency because, although DNS uses caching extensively to avoid latency, many CDNs use very short DNS cache lifetimes to give frequent opportunities for DNS-based load balancing and replica selection [14]. Latency is less critical at the DNS root [25], but unnecessary delay should be avoided [7]. Because of public attention to DNS latency, low latency is a selling point for many commercial DNS operators, many of whom deploy extensive distributed systems with tens, hundreds, or more than 1000 sites [8].

DNS deployments often use *IP anycast* [33,47] to reduce latency for clients. A *DNS service* is typically provided by two or more *authoritative DNS servers* [22], each defined in DNS on a separate IP address with an NS record [35]. With IP anycast, the IP address assigned to the authoritative DNS server is announced from many physically distributed *sites*, and BGP selects which clients go to which site—the anycast *catchment* of that site. DNS clients often select the lowest-latency authoritative server when they have a choice [40,43]. We show (Sect. 5) that this preference shifts client traffic to sites with the lowest latencies. (Although we focus on anycast, our approach can also be used to evaluate multiple unicast services serving the same zones).

DNS latency has been extensively studied [9,42,59]. Previous studies have looked at both absolute latency [59] and how closely it approaches speed-of-light optimal [28,63]. Several studies measure DNS latency from measurement systems with distributed vantage points such as RIPE Atlas [53], sometimes to optimize latency [7,34]. Recent work has shown how to measure anycast catchments with active probes with Verfploeter [12,13], and there is ongoing work to support RTT measurements. However, approaches to measure latency provide mixed coverage: large hardware-based measurements like RIPE Atlas only have about 11k active vantage points and cover only 8670 /24 IPv4 network prefixes [51] (May 2020), and commercial services have fewer than that. Verfploeter provides much better coverage, reaching millions of networks, but it depends on a response from its targets and so cannot cover networks with commonly-deployed ICMP-blocking firewalls. It is also difficult to apply to IPv6 since it requires a target list, and effective IPv6 target lists are an open research problem [16]. Finally, with the cost of active probing, Verfploeter is typically run daily and is too expensive to detect hourly changes (and RTT measurement support will require twice as much probing).

The main contribution of this paper is to fully evaluate and show operational results from *passive latency observations in DNS*. We show that passive observations of latency in TCP can provide continuous updates of latency with no additional traffic, providing operationally-useful data that can complement active probing methods such as Verfploeter or static observers such as RIPE Atlas. Such observations are not possible with DNS over UDP, and active probing is typically less frequent.

Observing latency from TCP, and in DNS, is not completely new, but prior work has not validated its accuracy and coverage. The TCP handshake has been used to estimate RTT at endpoints since 1996 [20], and it is widely used in passive analysis of HTTP (for example, [57]). Even in DNS, using TCP has been touched upon—the idea was shared with us by Casey Deccio, and `.cz` operators have used it in their service as described in non-peer-reviewed work that was independent from ours [31,32]. We validate that latency measured from UDP and our estimates from TCP match (Sect. 2.2). We show that DNS servers can choose to solicit TCP from selected clients to increase coverage, if they desire, with an implementation in Knot (Sect. 2.1).

Our second contribution is to show that *TCP-handshakes provide an effective estimate* of DNS latency. Although DNS most often uses UDP, leaving DNS-over-TCP (shortened to DNS/TCP) to be often overlooked, we show that there is enough DNS/TCP traffic to support good coverage of latency estimation. Prospecting through DNS traffic can find the latency "gold". Unlike prior approaches, passive analysis of TCP provides more coverage as busy clients send more queries, some with TCP. It provides good coverage of DNS traffic: for `.nl`, the top 100 ASes that send DNS/TCP traffic are responsible for more than 75% of *all* queries (Sect. 2.1), and we cover recursive servers sending the majority of queries. By scaling coverage with actual traffic, continuous passive RTT estimation can *increase temporal coverage* beyond current active approaches. For `.nl`, we cover 20k ASes every hour (Sect. 2.1). Finally, passive analysis is the only approach that provides good coverage for IPv6 networks, overcoming the problem of active probing with stateless IPv6 addresses [46].

Our final contribution is to show that *TCP-based latency estimation matters*—it detects latency problems in operational networks, improving *latency engineering* in anycast (Sect. 4). We identify *DNS polarization* as a problem that occurs when an Internet "hypergiant" [17,27,48] with a global footprint sends traffic over their own backbone to a single anycast location rather than taking advantage of an existing global anycast service. We show the importance of detecting and correcting this problem, reducing latency inflation by 150 ms for many clients of Google and Microsoft as they access `.nl` ccTLD and two commercial DNS providers. We have instrumented our open-source `ENTRADA` [61,70] with DNS/TCP RTT analysis. We provide a new tool, `Anteater`, that analyzes DNS/TCP RTT continuously to detect errors and failures in real-time (we released it freely at [37]), and extend an existing DNS analysis tool (dnsanon). These tools have been operational for more than two years at SIDN, the Netherlands ccTLD (`.nl`) operator, and were deployed in March 2021 by the B-Root root DNS server. During that deployment, our tools have detected several problems. In one case, some users experienced large increases in RTT due to traffic from Europe going to an anycast site in Australia (Sect. 4.4).

Our tools are freely available, including our changes to Knot [10], dnsanon, `Anteater`, and `ENTRADA`. Part of our data is from public TLDs, so privacy concerns prevent making data public. Our analysis follows current ethical guidelines: we never associate data with information about specific individuals, and our analysis is part of improving operations.

2 DNS/TCP for RTT?

While UDP is the preferred transport layer for DNS, TCP support has always been required to handle large replies [6] and all compliant resolvers are required to use TCP when the server sets the TC (truncated) bit [35]. TCP has also always been used for zone transfers between servers, and now increasing numbers of clients are using TCP in response to DNSSEC [2], response-rate limiting [66], and recently DNS privacy [23].

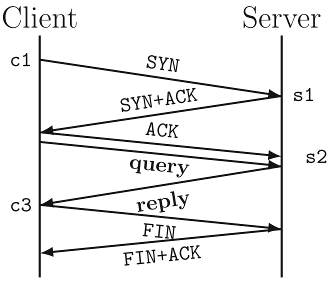


Fig. 1. TCP handshake and RTT measurements

The RTT between a TCP client and server can be measured passively during the TCP session establishment [20,34] or during the connection teardown [57]. In our work, we measure the RTT during the session establishment, as shown in Fig. 1: we derive the RTT between client and server by computing the difference between times $s2$ and $s1$, measured at the server. (In Sect. 2.2 we validate against client measurements of transaction time $c1$ and $c3$, which will be two RTTs (plus usually negligible server processing time)).

When we have multiple observations per target region (AS or prefix), we take the median. We choose median so that frequent retries will change the result, but occassional retries will not.

For passive TCP observations to support evaluation of anycast networks for DNS, (a) enough clients must send DNS over TCP so they can serve as *vantage points* (VPs) to measure RTT, and (b) the RTT for queries sent over TCP and UDP should be the same.

We next verify these two requirements, determining how many clients can serve as VPs with data from three production authoritative servers (Sect. 2.1) – two from the `.nl` zone, and `B-root`, one of the Root DNS servers [56]. We then compare the RTT of more than 8k VPs with both TCP and UDP to confirm they are similar (Sect. 2.2), towards two large anycast networks: `K` and `L-Root`, two of the 13 anycast services for the Root DNS zone.

2.1 Does TCP Provide Enough Coverage?

To assess whether DNS/TCP has enough coverage in production authoritative servers, we look at production traffic of two DNS zones: `.nl` and the DNS Root. For each zone we measure: (a) the number of resolvers using the service; (b) the number of ASes sending traffic; (c) the fraction of TCP queries the servers receive; (d) the percentage of resolvers using both UDP and TCP; and (e) the RTT of the TCP packets.

Our goal is to get a good estimate of RTT latency that covers recursive servers accounting for the majority of client traffic. If every query were TCP, we could determine the latency of each query and get 100% coverage. However, most DNS queries are sent over UDP instead of TCP.

Table 1. DNS usage for two authoritative services of `.nl` (Oct. 15–22, 2019).

	Queries		Resolvers		ASes	
	Anycast A	**Anycast B**	**Any. A**	**Any. B**	**Any. A**	**Any. B**
Total	5 237 454 456	5 679 361 857	2 015 915	2 005 855	42 253	42 181
IPv4	4 005 046 701	4 245 504 907	1 815 519	1 806 863	41 957	41 891
UDP	3 813 642 861	4 128 517 823	1 812 741	1 804 405	41 947	41 882
TCP	191 403 840	116 987 084	392 434	364 050	18 784	18 252
ratio TCP	5.02%	2.83%	21.65%	20.18%	44.78%	43.58%
IPv6	1 232 407 755	1 433 856 950	200 396	198 992	7 664	7 479
UDP	1 160 414 491	1 397 068 097	200 069	198 701	7 662	7 478
TCP	71 993 264	36 788 853	47 627	4 6190	3 391	3 354
ratio TCP	6.2%	2.63%	23.81%	23.25%	44.26%	44.85%

We, therefore, look for *recursive representation*—if we have a measured query over TCP, is its RTT the same as the RTTs of other queries that use UDP, or that are from other nearby recursive resolvers? If network conditions are relatively stable, the TCP query's RTT can represent the RTT for earlier or later UDP queries from the same resolver. Since /24 IPv4 prefixes (and /56 IPv6 prefixes) are usually co-located, DNS/TCP measurements from one IP can also represent other resolvers in the same prefix. Our goal is to find latency for DNS recursive resolvers, not all client networks—since recursive resolvers that generate the most traffic are most likely to send TCP queries, we expect good coverage even if TCP use is rare.

`.nl` Authoritative Servers. `.nl` currently (Oct. 2019) has four Authoritative DNS services, each configured to use IP anycast. We next examine data from two of these services. Anycast Services A and B employ 6 and 18 sites distributed globally. Each is run by a third-party DNS operator, one headquartered in Europe and the other in North America. They do not share a commercial relationship, nor do they share their service infrastructure.

DNS/TCP Usage: we analyze one week of traffic (2019-10-15 to -22) for each service using `ENTRADA`. That week from each service handles about 10.9 billion queries from about 2M resolvers spanning 42k Autonomous Systems (ASes), as can be seen in Table 1. The data shows that TCP is used rarely, accounting for less than 7% of queries for each anycast service. However, those queries represent more than a fifth of resolvers and 44% of ASes. (In all cases, TCP queries come from IP addresses that also send UDP queries).

AS Representation: We have TCP data for roughly 44% of all ASes (Table 1). This coverage is lower than we would prefer, but *these are the ASes that account for the majority of traffic*: the top DNS/TCP 10 ASes are responsible for half of all queries, while the top DNS/TCP 100 ASes account for 78% for Service A and 75% for B (Fig. 2). Although we miss many ASes, we next show we cover the prefixes in those ASes with recursive servers and we account for a large fraction of DNS traffic.

Traffic Coverage: We see that 5% of all queries are TCP, and they originate from about 20% of all resolvers (Table 1). While these are incomplete, we next show

Table 2. Traffic coverage for resolvers that use TCP in addition to UDP for DNS queries for `.nl` (Oct. 15–22, 2019).

	Anycast A	**Anycast B**
IPv4	4 005 046 701	4 245 504 907
TCP resolvers (DNS/UDP + DNS/TCP)	2 306 027 922	1 246 213 577
Ratio (%)	57.7%	29.35%
IPv6	1 232 407 755	1 433 856 950
TCP resolvers (DNS/UDP + DNS/TCP)	533 519 527	518 144 495
Ratio (%)	43.29%	36.13%

Table 3. DNS queries (in millions) for root DNS (E and G missing) – 2019-10-15 – 2019-10-22.

	A	**B**	**C**	**D**	**F**	**H**	**I**	**J**	**K**	**L**	**M**
Total	70601	40601	59033	88136	144635	31702	66582	115162	76761	105041	42702
IPv4	58552	33925	47675	74565	125020	25706	55874	96727	61378	88046	33687
UDP	56921	32334	45568	70969	118738	25234	51208	87891	60312	84059	31925
TCP	1631	1591	2107	3596	6282	472	4665	8836	1065	3986	1762
Ratio TCP	2.87%	4.92%	4.62%	5.07%	5.29%	1.87%	9.11%	10.05%	1.77%	4.74%	5.52%
IPv6	12049	6675	11357	13571	19614	5995	1070	18435	15383	16994	9014
UDP	11659	6280	10966	13071	18919	5825	936	15511	15108	16576	8268
TCP	389	394	391	499	694	169	1342	2923	274	418	746
Ratio TCP	3.34%	6.29%	3.57%	3.82%	3.67%	2.92%	14.34%	18.84%	1.82%	2.52%	9.03%

that they cover the majority of DNS traffic. In Table 2, we see that 29–58% of the total traffic (depending on IP version and anycast service) is from resolvers that have sent some TCP. As such, we have latency for at least 29% and up to 58% of DNS traffic. In addition, if we want full coverage, we describe below how we can induce coverage when it is necessary.

Root DNS: To confirm that DNS/TCP provides coverage beyond `.nl`, we also look at how many TCP queries are seen at most Root DNS servers [56] over the same period. Table 3 shows RSSAC-002 statistics [24,69] from 11 of the 13 Root DNS services reporting at this time. We see the ratio of TCP traffic varied for each service (known as "letters", from A to M) and IPv4 or IPv6, overall ranging from 2.8 (A Root over IPv4) up 18.9% (J Root over IPv6). This data suggests the root letters see similar DNS/TCP rates as `.nl`.

Inducing Coverage. While TCP coverage is not complete, we can get complete coverage by actively managing traffic to induce occasional TCP queries, as is often done in web systems (for example, [58]). The DNS specification includes the TC ("truncated") bit to indicate a truncated reply that must be retried over TCP. DNS Receiver Rate Limiting [66] (RRL) uses this mechanism to force possible UDP-based address spoofers to resend their queries with TCP. Switching to TCP allows TCP cookies to prevent spoofing [15].

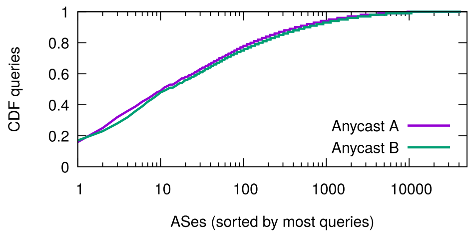


Fig. 2. `.nl`: queries distribution per AS.

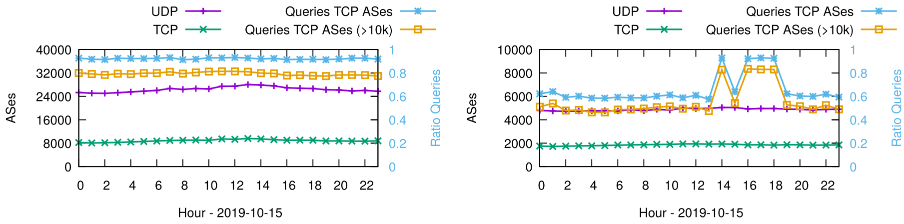
A DNS server can use this same mechanism to solicit TCP queries from selected clients, allowing us to determine RTTs. We have implemented this capability in the Knot DNS server [11], building on Knot's RRL implementation. Our implementation tracks each block (/24 IPv4 prefix, or /56 IPv6 prefix). When a UDP request from that block arrives, if there are insufficient TCP queries in the last hour, it returns an answer with the TC bit set with some probability. The probability of not setting the bit and the required number of RTT observations per hour are both configurable. (However, our measurements pre-date and so do not use this mechanism).

The cost of forcing TCP is two additional round trips, and some resolvers fail to convert to TCP [41]. TCP solicitation should therefore be used sparingly, although other deployed systems diverting some traffic to identify service problems (for example, Facebook [58]). To ensure the server's increased TCP load is negligible, RTT induction should be configured to balance better measurements against available computational resources.

Temporal Coverage. Next we investigate how much *temporal coverage* passive analysis of DNS/TCP provides. We require TCP connections to observe latency in each time period with confidence, so traffic rate per AS determines our temporal precision. We hope traffic allows temporal precision of 0.5 to 4 hours so passive analysis can support near-real-time monitoring over the day (Sect. 4.4).

To evaluate the number of TCP queries per AS in a given time interval, we analyze `.nl` traffic from Anycast A and B. We single out one day of traffic (the first day of Table 1, 2019-10-15). On this day, Anycast A and Anycast B received UDP queries from ∼37k ASes over IPv4, and from ∼6.4k ASes over IPv6 (notice that numbers in Table 1 are higher given they cover the whole week).

To evaluate how many ASes report enough data to estimate RTTs each hour, Fig. 3 shows TCP queries per hour for Anycast A. As a baseline, IPv4 (Fig. 3a) sees about 26.3k ASes that send UDP queries per hour (IPv4), and 4.8k for IPv6. Of these, about 8.8k also send TCP queries (1.8k for IPv6), allowing some IPv4 RTT information about 33% of ASes and for IPv6, 38% of ASes. However, these ASes that *also* send TCP queries are responsible for the majority of *all* queries (blue line in Fig. 3): more than 90% of IPv4 queries and more than 60% of all IPv6 queries. If we only consider ASes that send *at least* 10k TCP queries/hour, we still account for most of the traffic (yellow line in Fig. 3).



(a) Anycast A: IPv4 (b) Anycast A: IPv6

Fig. 3. `.nl` temporal coverage for Anycast A (Color figure online)

We conclude that a large number of ASes can be measured every hour with DNS/TCP. (We repeated the same analysis Anycast B (Appendix A) and also for another day 2019-10-21, both of them hold the same results [39]).

Summary: We see that TCP data provide good operational coverage and great temporal coverage. More importantly, TCP provides the *only* insight into IPv6 latency, since current active methods do not generalize to IPv6.

2.2 DNS/UDP vs. DNS/TCP RTT

We expect round-trip times measured with DNS/TCP and DNS/UDP to be similar. Next, we investigate that assumption.

We can compare DNS/UDP and DNS/TCP RTTs by comparing *query response times* and accounting for TCP connection setup. DNS/UDP makes a direct request and gets a response, while in DNS/TCP we set up the TCP connection (with a SYN–SYN/ACK handshake), so a TCP DNS request should take two RTTs (assuming no connection reuse, TCP fast-open, or other optimizations). We expect similar RTT estimates after dividing by two to account for TCP's handshake.

To confirm this claim we measure DNS/UDP and DNS/TCP query response times using RIPE Atlas [52]. Atlas provides about 11k devices in different locations around the world, allowing us to test many network conditions. As *targets*, we evaluate two large, globally distributed, production and public DNS anycast networks: `L-Root`, with 167 anycast sites, and `K-Root`, with 79 sites, which are two of the thirteen authoritative servers for the Root DNS zone. To measure DNS/UDP latency from probes to these root letters, we leverage existing measurements that run continuously on Ripe Atlas, every 4 min.

We then create DNS/TCP measurements towards `L` and `K-Root` (`[k,l]root-tcp` in [50]), within daily limits on query for RIPE Atlas. We study about 8.6k probes, running every 8 minutes (twice the interval of UDP measurements) for 24 hours, with results in Table 4. In these measurements, each Atlas probe directly queries the IPv4 address of the `K` and `L-Root`, without using a recursive resolver. For a fair comparison, we consider only probes that are present in both UDP and TCP measurements ($\cap$ Probes): 8.5k and 8.9k for `K`

Table 4. DNS/UDP vs. DNS/TCP Atlas measurements. Datasets: [50].

	K-Root		L-Root	
	UDP	TCP	UDP	TCP
Date	Sept 4–5, 2020		Sept 5–6, 2020	
Freq.	4 min	8 min	4 min	8 min
Probes	10608	8680	10595	8999
∩ Probes	8577		8901	
Queries	3759080	1516801	3756572	1600283
∩ Queries	3044067	1499078	3160277	1583243

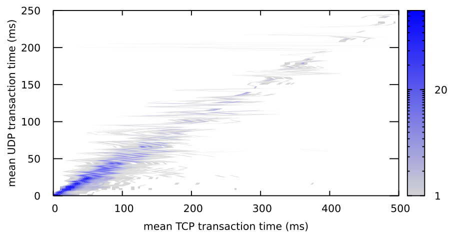


Fig. 4. `L-Root`: Density plot (log-scale) of number of Atlas observers with mean UDP and TCP DNS transaction times.

and `L-Root`, respectively. In the same way, we only evaluate queries from these matching probes: ~3M UDP queries and ~1.5M for TCP measurements, for each Root Letter (∩ Queries). Both measurements consider retries: UDP in the application and TCP by the kernel. In total, we analyze 9.2M queries for both letters. Then, for each Atlas probe, we compute its latency distribution.

Figure 4 shows a density plot of the number of RIPE Atlas observers with a given combination of mean DNS/UDP and DNS/TCP transaction times. Each combination is the mean of around 360 observations, and we report densities as log scale across the 8.9k Atlas probes available during the measurement. (We omit a handful of outliers with UDP means more than 250 ms.) We see a strong trend on the diagonal with a 1:2 UDP:TCP ratio, corresponding with TCP requiring two round trips. Slight variations from the diagonal represents queueing; considerable variation suggests experiments with retries. We see similar results (not shown due to space) for K-Root, and for median, and for 90%ile RTTs (see [39] for CDFs). The data shows a strong correlation between UDP and TCP, but also some outliers in the lower-left corner due to retransmission.

We can quantify similarity by computing the correlation coefficient of median UDP and half the median: the correlation is 0.913 for K-Root and 0.930 for L-Root. We also did a Student's *t*-Test, evaluating the hypothesis that the UDP mean and half the TCP mean are statistically identical with a 95% confidence. This test could not be rejected the majority of the time (64% of the time, in 5558 cases for K-Root and 5733 cases for L-Root), suggesting the results were often indistinguishable. Manual examination shows outliers are common in the cases where the hypothesis is rejected, suggesting a TCP-level retransmission. Although retransmission detection is possible, our results show usability even when minimizing computational requirements so as to optimize for low-overhead, real-item deployments. This experiment proves that passively observed TCP RTTs often provide a good representation of the RTTs that DNS/UDP will see.

3 Prioritizing Analysis

We have shown that DNS/TCP can be mined to determine RTTs (Sect. 2). Operational DNS systems must serve the whole world, there are more than 42k active ASes sending DNS queries to authoritative servers. Both detection and resolution of networking problems in anycast systems is labor intensive: detection requires both identifying specific problems and their potential root causes. Problem resolution requires new site deployments or routing changes, both needing human-in-the-loop changes involving trouble tickets, new hardware, and new hosting contracts.

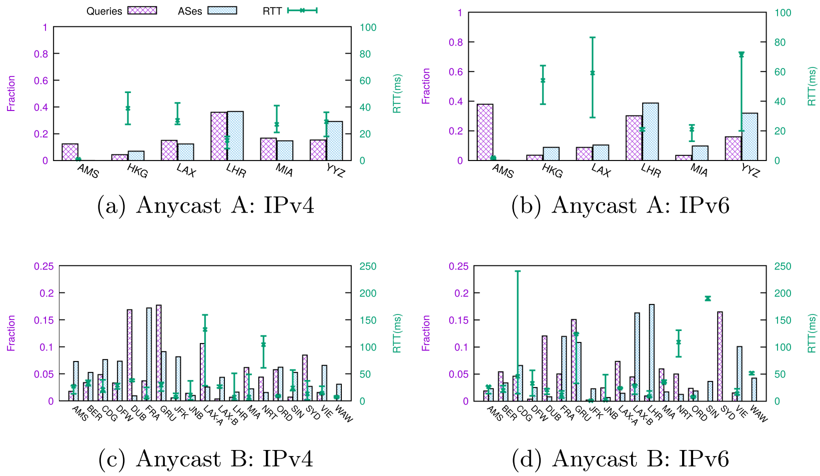
Overview: We use two strategies to *prioritize* the analysis of problems that are most important: per-anycast site analysis and per client AS analysis, and rank each by median latency, interquartile range (IQR) of latency, and query volume.

We focus on anycast sites because that part of the problem is under operator control. If we find sites with high latency, we can examine routing and perhaps correct problems [7,68].

Clients ASes examine the *user* side of the problem (at recursive resolvers), since client latency is a goal in DNS service. While performance in client ASes can be difficult to improve because we do not have a direct relationship with those network operators, we show in Sect. 4 that we can address problems in some cases.

Finally, we consider median latency, interquartile range, and query volume to prioritize investigation. Median latency is a proxy for overall latency at the site. The interquartile range (the difference between 75%ile and 25%ile latencies), captures the *spread* of possible latencies at a given site or AS. Finally, query volume (or rate) identifies locations where improvements will affect more users. We sort by overall rate rather than the number of unique sources to prioritize large ASes that send many users through a few recursive resolvers (high rate, low number of recursive IPs).

Prioritization by Site: Figure 5 shows per-site latency for `.nl`, broken out by protocol (IPv4 and IPv6) and site, for two anycast services (A and B). For each site, we show two bars: the fraction of total queries and the number of ASes (filled and hatched bar in each cluster). We overlay both with whiskers for latency (with median in the middle and 25%ile and 75%ile at whisker ends). In these graphs some sites (such as CDG for Anycast B in IPv6) stand out with high interquartile ranges, while others with lower interquartile range (for Anycast B, LAX-A and NRT in IPv4 and NRT and GRU in IPv6). We look at these cases in detail in Sect. 4.



(a) Anycast A: IPv4 (b) Anycast A: IPv6

(c) Anycast B: IPv4 (d) Anycast B: IPv6

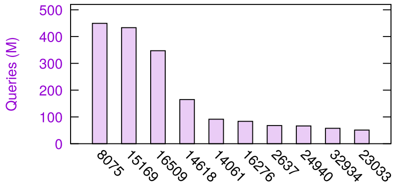
Fig. 5. `.nl` distribution of queries and ASes per site (pink bars) and latency (median, 25%ile, and 75%ile, (green lines), for each anycast site, for two services (Anycast A and B) and two protocols (IPv4 and IPv6). Data from 2020-10-15 to -22. (Color figure online)

Prioritization by Client AS: Figure 6 and Fig. 19 (Appendix B) show the latency distribution for the ten ASes with the largest query volume for Anycast A and B of `.nl`. (Due to space constraints, we show the complete list of AS names in Appendix C). While many ASes show good latency (low median and small interquartile range), we see the top two busiest ASes for Anycast A in IPv4 (Fig. 6a) show a high median and large interquartile range (Fig. 6b). These ASes experience anycast polarization, a problem we describe in Sect. 4.3.

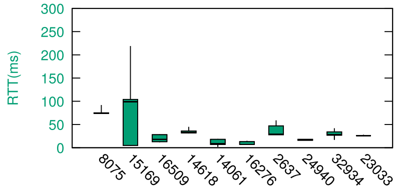
Figure 7 shows latencies for the top ASes for `B-root`, with quartile ranges as boxes and the 10%ile and 90%iles as whiskers. Rather than split by protocol, here we show both rankings and by query rate (Appendix B) on the x-axis. While rank gives a strict priority, showing ASes by rate helps evaluate how important it is to look at more ASes (if the next AS to consider is much, much lower rate, addressing problems there will not make as large a difference to users). We identify specific problems from these graphs next.

4 Problems and Solutions

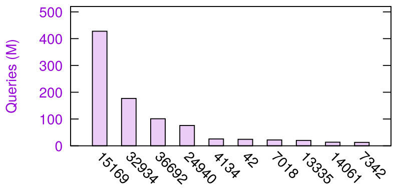
Given new information about IPv4 and IPv6 latency from DNS/TCP (Sect. 2), and priorities (Sect. 3), we next examine anycast performance for two of the four anycast services operating for `.nl`, and for `B-root`. For each problem, we describe how we found it, the root causes, and, when possible, solutions and outcomes. We show two problems that have been documented before (Sect. 4.1 and Sect. 4.2) and a new problem (Sect. 4.3). While some of these problems may have been eventually discovered when users encountered them, we have discovered



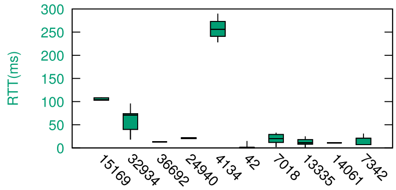
(a) Anycast A – IPv4 – Queries



(b) Anycast A – IPv4 – RTT



(c) Anycast A – IPv6 – Queries



(d) Anycast A – IPv6 – RTT

Fig. 6. `.nl` Anycast A queries and RTT for the top 10 ASes ranked by most queries (bars left axis). Data; 2019-10-15 to -22. ASes list on Appendix C).

each from our prioritized monitoring of DNS/TCP latency (Sect. 3). Our near-real-time monitoring (Sect. 4.4) supported the discovery of these problems before user complaints.

4.1 Distant Lands

The first problem we describe is *distant lands:* when a country has no anycast server locally and has limited connectivity to the rest of the world. When trans-Pacific traffic was metered, these problems occurred for Australia and New

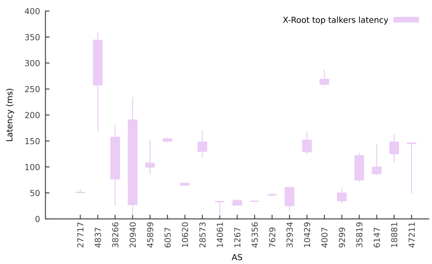


Fig. 7. `B-root`: latency of top talkers by rank

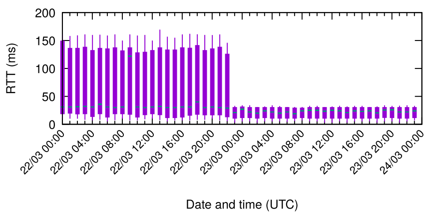


Fig. 8. Anycast B and Comcast: RTT before and after resolving IPv6 misconfiguration.

Zealand. Today we see this problem with China. China has a huge population of Internet users, but its international network connections can exhibit congestion [73].

Detection: We discovered this problem by observing large interquartile latency for `.nl`'s Anycast B in v4 (Fig. 5c) and v6 (Fig. 5d) at Tokyo (NRT, both v4 and v6), Singapore (SIN, v6), and CDG (v6), all with 75%iles over 100 ms.

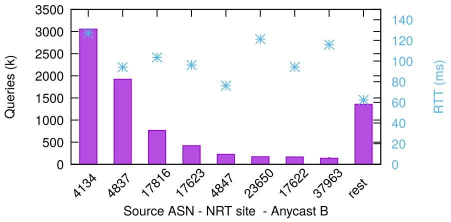


Fig. 9. Anycast B, Japan site (NRT): top 8 querying ASes are Chinese, and responsible for 80% of queries.

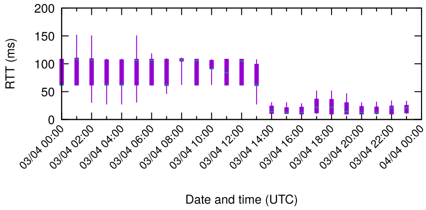


Fig. 10. `.nl` Anycast A and Microsoft (IPv4): RTT before and after depolarization.

These wide ranges of latency prompted us to examine which recursive resolvers visiting these sites and showed high latency. Many queries come from ASes in Asia (Fig. 9). NRT sees many queries (6.1% of total, more than its "fair share" of 5.2%). Of the top 10 ASes sending queries to NRT, 9 are from China (see Fig. 9).

We see many Chinese ISPs that also send IPv6 traffic to Paris (CDG), resulting in a large RTT spread. Not only must their traffic traverse congested, international links, but they also then travel to a geographically distant anycast site, raising the 75%ile RTT at CDG over 100 ms (even though its median is under 22 ms).

Resolution: While we can diagnose this problem, the best resolution would be new anycast servers for Anycast B inside China (or manipulate BGP to attempt to steer traffic to nearby Tokyo or Singapore sites). The operator is considering deployment options, but foreign operation of sites in China has only been recently allowed [73], and anycast operation from China risks service for some non-Chinese clients traversing the Chinese national firewall.

4.2 Prefer-Customer to Another Continent

The second root-cause problem we found is when one AS prefers a distant anycast site, often on another continent, because that site is a customer of the AS. (Recall that a common BGP routing policy is *prefer customer:* if an AS can satisfy a route through one of its customers, it prefers that choice over an alternate route through a peer or transit provider. Presumably, the customer is paying the AS for service, while sending the traffic to a peer or via transit is either cost-neutral or incurs additional cost).

We have seen this problem in two situations: at `.nl` Anycast B's Brazil site, and `B-root` for its site in South America. While the ISP should be able to choose where it sends its traffic, anycast service operators would like to know when such policies result in large client latencies, so that can consider exploring peering options that might lower latency.

`.nl` **Detection:** We detected this problem for `.nl` Service B by observing high IPv6 median latency (124 ms) for queries is in São Paulo, Brazil (GRU) in Fig. 5d. Examination of the data shows that many of the high-latency queries are from Comcast (AS7922), a large U.S.-based ISP. As with China and CDG, this case is an example of queries traveling out of the way to a distant anycast site, ignoring several anycast sites already in North America. We confirmed that North American clients of this AS were routing to the Brazil site by checking CHAOS TXT queries [1] from RIPE Atlas probes to Anycast B (data: ComcastV6 [50]).

`.nl` **Resolution:** We contacted `.nl` Anycast B's operator, who determined that the issue was with one of their upstream providers. This provider had deployed BGP communities to limit the IPv4 route to South America. After our contact, they deployed the same community for IPv6, and the Comcast traffic remained in the US.

We first confirm the problem was resolved by analyzing traces from Anycast B, and by confirming that Comcast IPv6 clients were now answered by other North American sites. The solution reduced 75%ile latency by 100 ms: in Fig. 8 before the change, IPv6 shows IQR of 120 ms for Anycast B. After this change on 2020-03-23t00:00, we see the IQR falls to 20 ms. Second, we also verified with Atlas probes hosted on Comcast's network (data: ComcastV6-afterReport in [50]), and the median RTT from Comcast Atlas was reduced from 139 ms to 28 ms.

`B-root` **Detection:** `B-root` has observed high latencies for traffic going to a South-American anycast site of `B-root`. As with `.nl` and GRU, we examined traffic and identified a primarily-North American ISP that was sending all of its traffic to the South American site, ignoring all other lower-latency sites. We then confirmed that an AS purchases transit from this ISP.

`B-root` **Resolution:** We have not yet a completely satisfactory resolution to this problem. Unfortunately, the AS that purchases transit from the North American ISP does not directly peer with `B-root`, so we cannot control its peering. We currently poison the route to prevent latency problems, which significantly reduces traffic arriving at this site.

4.3 Polarization with Google and Microsoft

We next describe *anycast polarization*, a problem we first described in June 2020 [39]. We are the first to explain and demonstrate the impact of polarization on performance, although subsequent work reported it in a testbed study [64]. Like prefer-customer, it involves high latency that derives from traffic being

Table 5. Anycast A: Polarized ASes and query distribution (Oct 15-22,2019).

	Queries	Queries top site	(% top site)
Google	860 775 677	860 774 158	99.9998
IPv4	433 145 168	433 145 119	99.9999
IPv6	427 630 509	427 629 039	99.9997
Microsoft	449 460 715	449 455 487	99.9988
IPv4	449 439 957	449 434 729	99.9988
IPv6	20 758	20 758	100

needlessly sent to another continent. But it follows from BGP's limited knowledge of latency (AS path length is its only distance metric) and the flattening of the Internet [27].

Detecting the Problem. We discovered this problem by examining DNS/TCP-derived latency from the top two ASes sending queries to `.nl` Anycast A. As seen in Fig. 6b and Fig. 6d, AS8075 (Microsoft) and AS15169 (Google) show very high IPv4 median latency (74 ms and 99 ms), and Google shows a very high IQR (99 ms) Google also shows a high IPv6 median latency (104 ms).

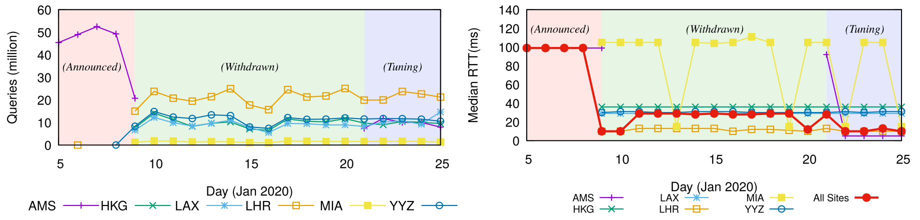
Both Google and Microsoft are "hypergiants", with data centers on multiple continents (for `.nl`, ∼85% of Google's traffic is from its Public Resolvers [38,62]). Both also operate their own international backbones and peer with the Internet in dozens of locations. These very high latencies suggest much of their DNS traffic is traveling between continents and not taking advantage of `.nl`'s global anycast infrastructure. This problem occurs in hypergiants with backbones that do not consider multiple exits and anycast—by default they will route all their traffic to one global anycast size, creating polarization. For companies with islands connected by transit providers (without a corporate backbone), each island will compute routing locally, so anycast "just works".

Confirming the Problem: `.nl` Anycast A has six sites, so we first examine how many queries go to each site. Table 5 shows the results—all or very nearly all (four or five "nines") go to a single anycast site due to routing preferences. For Google, this site is in Amsterdam, and for Microsoft, Miami.

While a preferred site is not a problem for a small ISP in one location, it is the root cause of very high latency for these hypergiants, who often route global traffic internally over their own backbones, egressing to one physical location. Even if it is the best destination for some of their traffic, one location will not minimize latency for multiple, globally distributed, data centers. Such routing defeats latnecy advantages of distributed anycast deployment [43,59].

Depolarizing Google to `.nl` Anycast A. *Root-cause:* We first investigated Google's preference for AMS. `.nl` directly operates the AMS site (the other 5 sites are operated by a North American DNS provider). We determined (working with both the AMS and Google operators) that Google has a direct BGP peering with the site at AMS. BGP prefers routes with the shortest AS-PATH, and in addition, ASes often prefer Private Network Interconnect (PNIs) over equal length paths through IXPs, so it is not surprising it prefers this path. (The general problem of BGP policy interfering with the lowest latency is well documented [4,5,7,28,34,59]. We believe we are the first to document this problem with hypergiants and anycast through PNI).

We next describe how we worked with the AMS operators and Google to resolve this problem. We document this case as one typical resolution to show the need for continuous observation of DNS latency through DNS/TCP to find the problem and confirm the fix.



(a) IPv4 - Queries (b) IPv4 - RTT (ms)

Fig. 11. `.nl` Anycast A: queries and median RTT per site from Google (AS15169) – January 2020.

Figure 11 show the effects of our traffic engineering on anycast use and query latency for IPv4 (IPv6 figures in Appendix D). Each graph shows traffic or median client latency for each of the 6 `.nl` Anycast A sites. (Query latency is determined by DNS/TCP traffic over each day.) The graphs show behavior over January 2020, January 5th to 9th (the left, pink area) before any changes, the 9th to the 21st (the middle, green area) when the AMS route was withdrawn, and finally after the 21st (the right, blue region) when AMS was restored, but with different kinds of policy routing.

These graphs confirm that initially, AMS received all traffic from Google, causing Anycast A to appear to be a *unicast* service to Google, precluding locality in a global anycast service. We see that the median latency for Google is about 100 ms (Fig. 6a). Withdrawing the AMS peering with Google corrects the problem, spreading queries across multiple anycast sites, benefiting from geographic locality. Median latency drops to 10 to 40 ms, although still around 100 ms at YYZ in Miami for IPv4. LHR is now the busiest site, located in Europe.

Use of the North American sites considerably lowers median latency. We show in Fig. 12 the depolarization results for all sites combined, for IPv4 and IPv6. For both IPv4 and IPv6, we see median latency for all sites combined reducing 90 ms, from 100 to 10ms. The IQR was reduced from 95 to 10 ms for IPv4. For

Table 6. BGP manipulations on AMS site of Anycast A – IPv4 and IPv6 prefixes to Google (AS15169) on Jan 21, 2020 (Time in UTC). NE: No Export

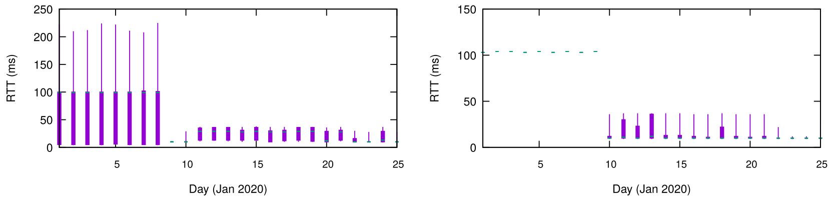
Op.	Day	Time	Prepend	Community	AMS(%)
1	21	15:00	2x	–	>0
2	22	9:53	2x	NE	>0
3	22	9:59	1x	–	100
4	22	10:21	1x	NE	100
5	22	10:37	1x	NE,15169:13000	100
6	22	11:00	2x	NE	>0

IPv6, we observed few queries over TCP between Jan. 1 and 9, so they are not representative. After depolarizing, we see more queries over TCP.

Although overall latency improves, omitting the AMS site misses the opportunity to provide better latency to their data centers in the Netherlands and Denmark. We therefore resumed peering over the BGP session, experimenting with several policy routing choices shown in Table 6. We experimented with 1x and 2x AS-PATH prepending, no-export, and a Google-specific "try-not-to-use this path" community string [18]. We found that no-export and the community string had no effect, perhaps because of the BGP session, and neither did single prepending. However, double AS-PATH prepending left AMS with about 10% of the total traffic load. Full details of our experiments are in our technical report [39].

Depolarizing Microsoft to `.nl` Anycast A. Detection: We discovered Microsoft anycast polarization through analysis of DNS/TCP across ASes (Fig. 6b and Fig. 6d) AS8075 (Microsoft) and AS15169 (Google) Microsoft's preferred site for `.nl` Anycast A is Miami (MIA), a different preference than Google's, but the outcome was the same: large latency (median 80 ms) because global traffic goes to one place.

Resolution: Again, we worked with the operators at `.nl` Anycast A MIA and Microsoft to diagnose and resolve the problem. We confirm that Anycast had



(a) IPv4 (b) IPv6

Fig. 12. Google depolarization results and RTT.

a peering session with Microsoft in MIA, and not at any other sites. Again, the result was a short AS-PATH and a preference for all Microsoft data centers to use the Microsoft WAN to this site rather than other `.nl` Anycast A anycast sites (having different upstream providers per anycast site may cause such traffic distributions [34]).

Options that could mitigate this polarization include de-peering with Microsoft in MIA, peering with Microsoft at the remaining sites, or possibly BGP-based traffic engineering. Because our ability to experiment with BGP was more limited at this site, and we could not start new peerings at other sites, the operator at MIA de-peered with Microsoft at our recommendation.

Figure 10 shows latency for this AS before and after our solution. Removing the direct peering addressed the problem, and Microsoft traffic is now distributed across all `.nl` Anycast A sites. As a result, the IQR falls from about 80 ms to 13 ms. The median latency also falls by 70 ms, from 90 ms to 20 ms. Our technique identifies problems with polarization and shows the dramatic improvement that results.

4.4 Detecting BGP Misconfiguration in Near Real-Time

Because it poses no additional cost on the network, passive measurement of anycast latency with DNS/TCP is an ideal method for *continuous, on-the-fly* detection of BGP misconfiguration. To this end, we have developed and deployed `Anteater` within `.nl`, which is a *live* monitoring system that retrieves DNS/TCP RTT continuously.

We show the `Anteater` architecture in Fig. 13. First, traffic is collected at authoritative DNS servers of `.nl`, which is then exported to ENTRADA [61,70], an open-source DNS traffic streaming warehouse that employs Hadoop [60] and continuously ingests `pcap` files from these servers. (We also use ENTRADA for other applications as well [36,67]). `ENTRADA` that extracts RTT for incoming TCP handshakes, making it available for queries using Impala [26], an open-source SQL engine for Hadoop. With Hadoop, `ENTRADA` supports scalable analysis of large traffic.

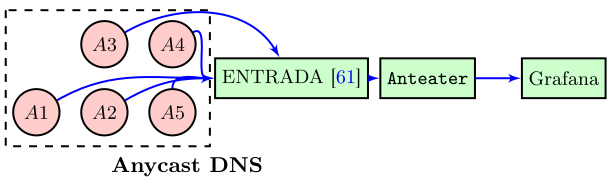


Fig. 13. `Anteater` monitoring at `.nl`

`Anteater` then retrieves DNS/TCP RTT data for each anycast server, anycast site, and various ASes, on an hourly basis (given that `.nl` authoritative servers have good temporal coverage in 1h frames Sect. 2.1). It stores the information

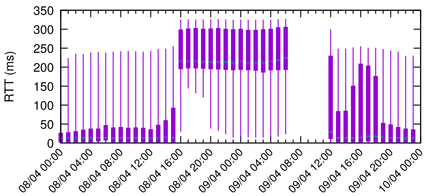


Fig. 14. Anycast B SYD site: IPv4 latency.

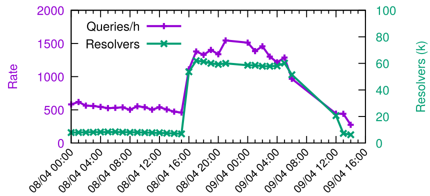


Fig. 15. Anycast B SYD: queries rate and resolvers.

on a relational database (PostgreSQL). We then use Grafana [19], a data visualization dashboard and alert system. We then configure RTT thresholds that triggers Grafana to send `.nl` operators email alerts. `Anteater` has been used in `.nl` for the past two years and its proven helpful in detecting BGP misconfigurations. We have released `Anteater` at [37]. Next we illustrate this use-case with one example from that deployment.

EU Traffic Winding up in Australia: On 2020-04-08, `.nl` operators received an alert from `Anteater` that detected a jump in median DNS RTT for Anycast B, from 55 ms to more than 200 ms (see Fig. 14) but only for IPv4 traffic, and not for IPv6.

To investigate this change, we evaluated the number of ASes (Fig. 14), resolvers, and query rates (Fig. 15) using `Anteater`'s Grafana dashboard. We see that the number of resolvers, queries, and latency grow, with many more ASes, and about 3× more queries and resolvers. To rule out DDoS attacks or a sudden burst in popularity for our domain, we confirmed that these ASes and resolvers have migrated from other sites (mostly Germany, site FRA) and went to SYD. Since many of these clients are in Europe, this nearly antipodal detour explains the latency increase.

We reached out to the operator of `.nl` Anycast B SYD. They confirmed and were already aware of the routing change. They informed us that a set of their SYD prefixes had accidentally to propagated through a large, Tier-1 transit provider. Since this provider peered with many other ASes in many places around the globe, their propagation of the Anycast B anycast prefix provided a shorter AS-Path and sent traffic to SYD. We also confirmed these routing changes on the RIPE RIS database of routing changes [54].

While catchment changes are not bad, route leaks that mis-route Europe to Australia are not an improvement. The lightweight nature of DNS/TCP observations of latency support 24 × 7 monitoring and allowed us to detect this problem, which is why we developed `Anteater` to monitor the `.nl` operations.

5 Anycast Latency and Traffic

While DNS/TCP can be used to discover anycast latency, does latency matter? DNS caching means *users* are largely insulated from latency. We next confirm

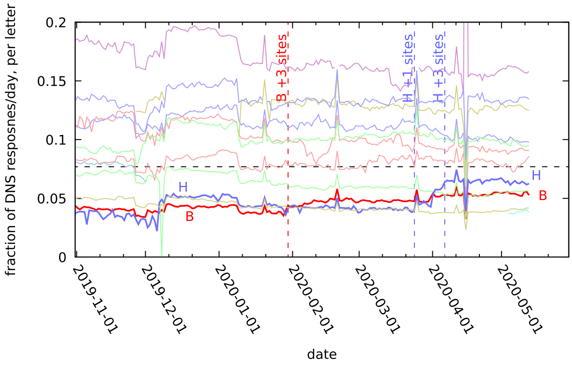


Fig. 16. Fraction of traffic going to each root anycast service, per day, from RSSAC-002 data. B- and H-Root are bold lines.

that latency does influence *traffic* to services when users have the choice of several. A causal relationship between client selection and latency was previously shown experimentally [43] and through code analysis [71], and our operational measurements adds operational measurement confirmation to these results.

Prior work has considered recursive resolver preference for lower latency [44]. Here we turn that analysis around and explore how changing anycast infrastructure shifts a client's preferences towards authoritative name servers. We confirm that lower latency results in increased traffic from recursive resolvers that have a choice between multiple anycast service addresses providing the same zone. (This question differs from studies that examine the optimality of a specific anycast service with multiple sites [28,29].)

To examine this question, we use public RSSAC-002 statistics for the root server system [56]. From this we use the "traffic-volume" statistic, which reports queries per day for each root anycast service. (Recall that the Root DNS is provided by 13 different anycast service addresses per IP version, each using a different anycast infrastructure.) We show 6 months of data here (2019-11-01 to 2020-05-31), but we have noticed similar trends since 2016. This analysis omits G- and I-Root, which did not provide data during this period.

Figure 16 shows the fraction of traffic that goes to each anycast service in the root server system for one year. Two root letters deployed additional sites over this period: B-Root originally had 2 sites but added 3 sites in 2020-02-01, then optimized routing around 2020-04-01. H-Root originally had 2 sites but deployed 4 additional sites on 2020-02-11 and 3 additional sites on 2020-04-06. While other letters also added sites, B and H's changes were the largest improvements relative to their prior size. We see that B and H's share rises from about 4% in 2019-11 to about 6% in 2020-05.

This data confirms that when new sites are created at a root letter, they offer some clients lower latency for that letter. Lower latency causes some clients to shift more of their traffic to this letter (automatically, as described in [43]), so its share of traffic relative to the others grows.

6 Related Work

Passive TCP Evaluation: Janey Hoe was the first to extract RTT from the TCP handshake [20], and several groups have used it since then (*e.g.*,, Facebook HTTP traffic [57]). We use this old idea, but we apply it to DNS RTT estimation and to use to engineer and monitor Anycast DNS services in near real-time. In a non-peer-reviewed work performed previously but independently from our own, `.cz` operators [31,32] also employed DNS/TCP RTT to evaluate latency from their services. While both use the same idea (derive latencies from the TCP handshake), ours provides a comprehensive validation (Sect. 2). We also act on the results, by carefully manipulating BGP to solve the identified problems, and reduce latency in up to 90% (Sect. 4). Besides, our work includes freely three tools: `dnsanon`, `Anteater`, and a modified version of KnotDNS. Linux `ss` and `ip` utilities [30] can be also used to retrieve TCP information such as RTT. However, they only provide *averages*. Although TCP congestion control may interact with latency, since DNS/TCP is usually short (a single query and reply), such interactions will be rare.

Anycast DNS Performance: Having a single upstream provider has been previously proposed as a solution to avoid routing unexpected behavior [4]. Later research evaluated the impact of *number of sites* and anycast performance, showing that, counterintuitively, sometimes more sites actually increase latency [59]. The behavior of anycast under DDoS has been examined [42], using data from the 2015 attacks against the Root DNS servers [55]. Our discovery of polarization in Google has been showin in subsequent testbed experiments [64]. We had already shared results of polarization for multiple hypergiants [39], and are the first to quantify the performance and show the benefits of BGP-based fixes.

There is one approach to measure anycast latency today: active measurements. RIPE Atlas [52] measures latency from about 11k physical devices distributed worldwide. Commercial services are known to have fewer vantage points. Our approach instead uses passive analysis of TCP traffic from real clients. It provides far better coverage than RIPE Atlas (Sect. 2.1). We expect that Verfploeter will soon support RTT measurements. Even when it does support RTT measurements, our approach can provide coverage for all networks interacting with the service. In addition, since our analysis is passive, it places no additional strain on other networks and can run 24×7. Last, a previous work proposed using new BGP communities to improve the site catchment, which, in turn, requires protocol level changes [28]. Contrary to their approach, ours relies only on passive TCP traffic and does not involve protocol changes.

Anycast Optimization for Large CDNs with Multiple Providers: Going beyond how many sites and where to place them, McQuistin *et al.* [34] have investigated anycast networks with multiple upstream providers, as is common for large CDNs. When different sites have different peers or transits catchment inconsistencies can result, as we saw with Google and Anycast A (Sect. 4.3). They propose taking active measurements of catchments each day and operator evaluation of catchment changes Our work also detects catchment changes, but only when it affects latency (see Sect. 4.3 and Sect. 4.3), fortunately when changes matte.r Schlinker *et al.* [57] describe how Facebook monitors their CDN for web content, detecting anycast latency problems for their users. Our work instead focuses shows how TCP results can summarize latency for a mostly UDP-based workload, and studies authoritative DNS traffic, from recursive resolvers (not end-users).

Performance-Aware Routing: Todd *et al.* [3] compare data from proposals for performance-aware routing from three content/cloud providers (Google, Facebook, and Microsoft) and show that BGP fares quite well for most cases. Others proposed to perform traffic engineering based on packet loss, latency and jitter [45,49].

DNS over TCP, TLS, and HTTP: There is recent interest in DNS over TCP and TLS [23,72] and HTTP [21] to improve privacy. Most such work emphasizes stub-to-recursives resolvers, while we focus on recursive-to-authoritative, where only now IETF is considering alternatives to UDP. Increased use of DNS over connection-oriented transport protocols will improve coverage we can provide.

7 Conclusions

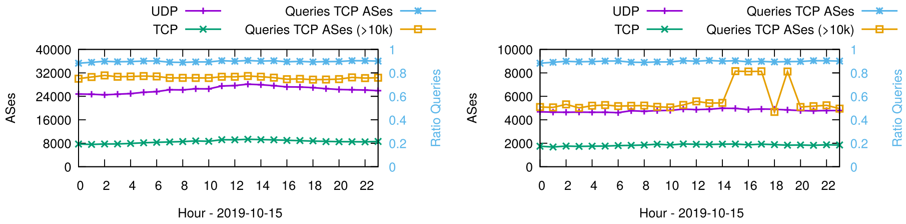
We have shown that DNS TCP connections are a valuable source of latency information about anycast services for DNS. Although TCP is not (today) the dominant transport protocol for DNS, we showed that there is enough DNS/TCP to provide good coverage for latency estimation. We also showed how we prioritize the use of this information to identify problems in operational anycast networks. We have used this approach to study three operational anycast services: two anycast servers of `.nl`, and one root DNS server (`B-root`). We documented one new class of latency problems: anycast polarization, an interaction where hypergiants get pessimal latency (100–200 ms) because of a poor interaction between their corporate backbones and global anycast services. We showed how we addressed this problem for `.nl`'s Anycast A with both Google and Microsoft. We also documented several other problems for anycast latency discovered through our analysis of DNS/TCP and showed that it enables continuous monitoring. Last, we release freely two tools (`dnsanon` and `Anteater`) and a modified version of KnotDNS. We believe this approach will be of use to other DNS operators.

Acknowledgments. We thank the operators of `.nl` Anycast A and B and `B-root` for their time and collaboration. We also thank Casey Deccio for first proposing using TCP hanshake to measure DNS latency. Finally, we thank Klaus Darilion and our paper's anonymous reviewers for their paper suggestions.

John Heidemann's research in this paper is supported in part by the DHS HSARPA Cyber Security Division via contract number HSHQDC-17-R-B0004-TTA.02-0006-I (PAADDOS) and by NWO. His and Wes Haradaker's research are also supported in part by NSF CNS-1925737 (DIINER), Giovane C. M. Moura, Joao Ceron, Jeroen Bulten, and Cristian Hesselman research in this paper is supported by the Conconrdia Project, an European Union's Horizon 2020 Research and Innovation program under Grant Agreement No 830927.

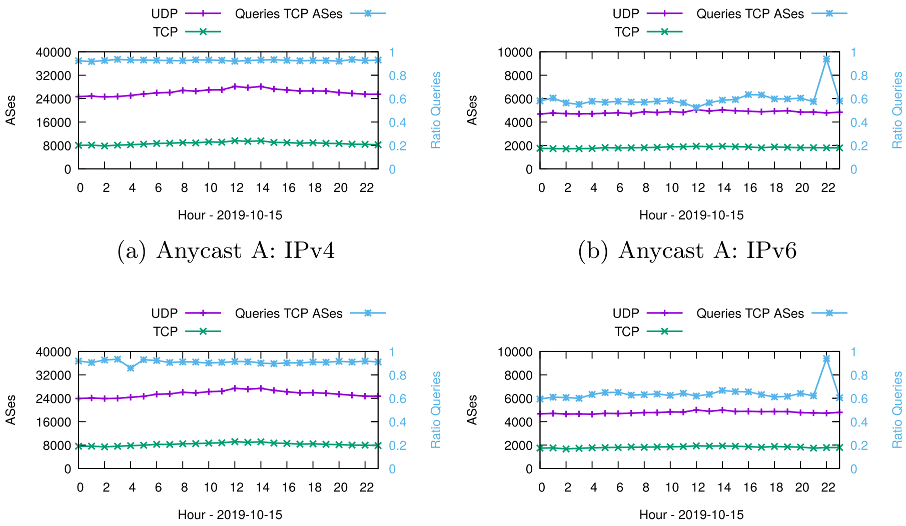
A Extra Graphs on Temporal Coverage

Figure 17 show the temporal coverage of Anycast B for `.nl`. Figure 18 shows temporal coverage for 2019-10-21. Together these graphs show that a core of many ASes are covered by TCP at all times of day.



(a) Anycast B: IPv4 (b) Anycast B: IPv6

Fig. 17. `.nl` temporal coverage for Anycast B



(a) Anycast A: IPv4 (b) Anycast A: IPv6

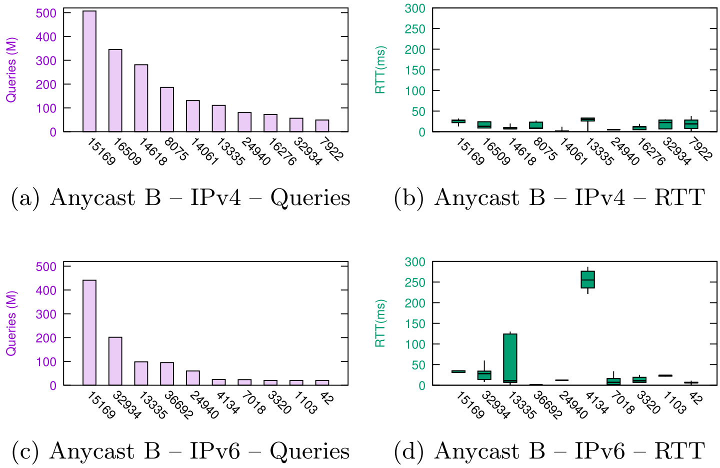
(c) Anycast B: IPv4 (d) Anycast B: IPv6

Fig. 18. `.nl` temporal coverage for Anycast A and B on 2019-10-21

B Anycast Extra Data

Figure 19 shows the the latency for the top 10 ASes of Anycast B.

Figure 20 shows the latency for `B-root` top talkers by data size (log scale).



(a) Anycast B – IPv4 – Queries (b) Anycast B – IPv4 – RTT

(c) Anycast B – IPv6 – Queries (d) Anycast B – IPv6 – RTT

Fig. 19. `.nl` Anycast B query RTT for the top 10 ASes ranked by most queries (bars left axis). Data: 2019-10-15 to -22.

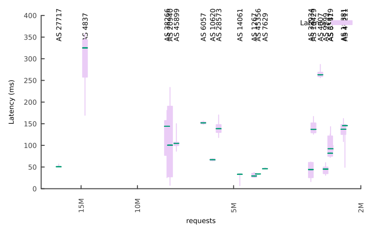
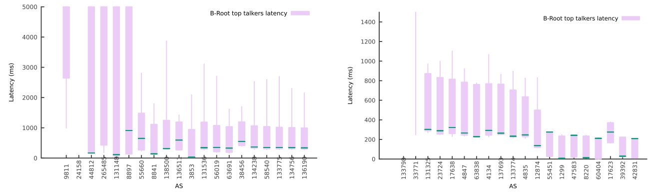


Fig. 20. `B-root` latency of top talkers by data size (log scale)

C Anycast A and B Top ASes

Table 7 shows ASes names and countries for top ASes observed for Anycast A and Anycast B, as shown in Fig. 6 and discussed in priorization (Sect. 3) (Fig. 21).



(a) `B-root` AS latency diversity for IPv4 by rank (b) `B-root` latency diversity for IPv6 by rank

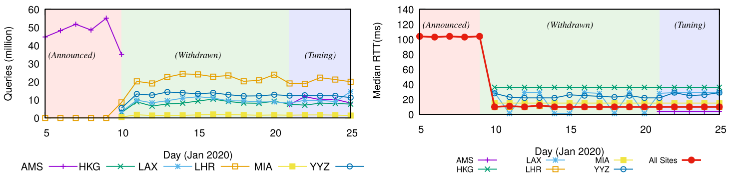
Fig. 21. Latency analysis to `B-root` by AS latency diversity.

Table 7. ASes in the top 10 lists of Anycast A and B

AS number	AS name	Country
42	WoodyNet (PCH)	US
1103	SURFNet BV	NL
2637	Georgia Institute of Technology	US
3320	Deutsche Telekom AG	DE
4134	China Telecom Backbone	CN
7018	AT&T Services, Inc.	US
7342	Verisign infrastructure and Operations	US
7922	Comcast	US
8075	Microsoft Corporation	US
13335	Cloudflare	US
14061	DigitalOcean LLC	US
14618	Amazon.com Inc.	US
15169	Google	US
16276	OVH SAS	FR
16509	Amazon.com Inc.	US
23033	Wowrack.com	US
24940	Hetzner Online GmbH	DE
32934	Facebook, Inc.	US
36692	Cisco OpenDNS, LLC	US

D Depolarizing Google: IPv6 Graphs

Figure 22 shows the IPv6 depolarization graphs for Anycast A and Google.



(a) IPv6 - RTT (ms) (b) IPv6 - RTT(ms)

Fig. 22. `.nl` Anycast A (IPv6): queries and median RTT per site from Google (AS15169) – January 2020.

References

1. Allman, M., Floyd, S., Partridge, C.: Increasing TCP's initial window. RFC 2414, IETF, September 1998
2. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS security introduction and requirements. RFC 4033, Internet Request For Comments, March 2005
3. Arnold, T., et al.: Beating BGP is harder than we thought. In: Proceedings of the 18th ACM Workshop on Hot Topics in Networks, HotNets 2019, pp. 9–16. Association for Computing Machinery, New York (2019)
4. Ballani, H., Francis, P.: Towards a global IP anycast service. In: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2005, pp. 301–312. Association for Computing Machinery, New York (2005)
5. Ballani, H., Francis, P., Ratnasamy, S.: A measurement-based deployment proposal for IP anycast. In: Proceedings of the 2006 ACM Conference on Internet Measurement Conference, IMC, pp. 231–244. ACM, October 2006
6. Bellis, R.: DNS transport over TCP–implementation requirements. RFC 5966, Internet Request For Comments, August 2010. https://www.rfc-editor.org/rfc/rfc5966.txt
7. Bellis, R.: Researching F-root anycast placement using RIPE Atlas. Ripe blog, October 2015
8. Calder, M., Fan, X., Hu, Z., Katz-Bassett, E., Heidemann, J., Govindan, R.: Mapping the expansion of Google's serving infrastructure. In: Proceedings of the ACM Internet Measurement Conference, pp. 313–326. ACM, Barcelona, October 2013
9. Castro, S., Wessels, D., Fomenkov, M., Claffy, K.: A day at the root of the internet. ACM Comput. Commun. Rev. **38**(5), 41–46 (2008)
10. Charnsethikul, P.: Knot extensions to support TCP soliciation, March 2021. https://ant.isi.edu/software/dnsrtt/
11. CZ.NIC: Knot DNS, November 2011. https://www.knot-dns.cz/
12. de Vries, W.B., Aljammaz, S., van Rijswijk-Deij, R.: Global scale anycast network management with verfploeter. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium. IEEE, Budapest, April 2020
13. de Vries, W.B., de O. Schmidt, R., Haraker, W., Heidemann, J., de Boer, P.T., Pras, A.: Verfploeter: broad and load-aware anycast mapping. In: Proceedings of the ACM Internet Measurement Conference, London, UK (2017)
14. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Weihl, B.: Globally distributed content delivery. IEEE Internet Comput. **6**(5), 50–58 (2002)
15. Eddy, W.: TCP SYN flooding attacks and common mitigations. RFC 4987, IETF, August 2007
16. Gasser, O., et al.: Clusters in the expanse: understanding and unbiasing IPv6 hitlists. In: Proceedings of the Internet Measurement Conference 2018, IMC 2018, pp. 364–378. Association for Computing Machinery, New York (2018)
17. Gigis, P., et al.: Seven years in the life of hypergiants' off-nets. In: Proceedings of ACM SIGCOMM 2021. Virtual Event, August 2021
18. Google: Google BGP communities, January 2020. https://support.google.com/interconnect/answer/9664829?hl=en
19. Grafana Labs: Grafana documentation, July 2020. https://grafana.com/
20. Hoe, J.C.: Improving the start-up behavior of a congestion control scheme for TCP. In: Proceedings of the ACM SIGCOMM Conference, pp. 270–280. ACM, Stanford, August 1996

21. Hoffman, P., McManus, P.: DNS Queries over HTTPS (DoH). RFC 8484, IETF, October 2018
22. Hoffman, P., Sullivan, A., Fujiwara, K.: DNS terminology. RFC 8499, IETF, November 2018
23. Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., Hoffman, P.: Specification for DNS over transport layer security (TLS). RFC 7858, Internet Request For Comments, May 2016
24. ICANN: RSSAC002: RSSAC Advisory on Measurements of the Root Server System, November 2014. https://www.icann.org/en/system/files/files/rssac-002-measurements-root-20nov14-en.pdf
25. Koch, T., Li, K., Ardi, C., Katz-Bassett, E., Calder, M., Heidemann, J.: Anycast in context: a tale of two systems. In: Proceedings of the ACM SIGCOMM Conference. ACM, Virtual, August 2021. https://www.isi.edu/
26. Kornacker, M., et al.: Impala: a modern, open-source SQL engine for Hadoop. In: Cidr. vol. 1, p. 9 (2015)
27. Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., Jahanian, F.: Internet inter-domain traffic. In: Proceedings of the ACM SIGCOMM Conference, pp. 75–86. ACM, New Delhi, August 2010
28. Li, Z., Levin, D., Spring, N., Bhattacharjee, B.: Internet anycast: performance, problems, & potential. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, pp. 59–73. Association for Computing Machinery, New York (2018)
29. Liang, J., Jiang, J., Duan, H., Li, K., Wu, J.: Measuring query latency of top level DNS servers. In: Proceedings of the International conference on Passive and Active Measurements, pp. 145–154. PAM, March 2013
30. Linux Foundation: networking:iproute2 [Wiki], March 2021. https://wiki.linuxfoundation.org/networking/iproute2
31. Andzinski, M.: Passive analysis of DNS server reachability, June 2019. https://centr.org/library/library/centr-event/rd14-andzinski-passive-analysis-of-dns-server-reachability-20190529.html
32. Andzinski, M.: Passive analysis of DNS server reachability, November 2019. https://www.nic.cz/files/nic/IT_19/prezentace/12_andzinski.pdf
33. McPherson, D., Oran, D., Thaler, D., Osterweil, E.: Architectural Considerations of IP Anycast. RFC 7094, IETF, January 2014
34. McQuistin, S., Uppu, S.P., Flores, M.: Taming anycast in the wild internet. In: Proceedings of the Internet Measurement Conference, IMC 2019, pp. 165–178. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3355369.3355573
35. Mockapetris, P.: Domain names - implementation and specification. RFC 1035, IETF, November 1987
36. Moura, G.C.M., Müller, M., Wullink, M., Hesselman, C.: nDEWS: a new domains early warning system for TLDs. In: NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium, pp. 1061–1066 (2016)
37. Moura, G.C.M.: Anteater, March 2021. https://github.com/SIDN/anteater
38. Moura, G.C.M., Castro, S., Hardaker, W., Hesselman, C.: Clouding up the internet: how centralized is DNS traffic becoming? In: Proceedings of the ACM Internet Measurement Conference, p. to appear. ACM, Virtual Conference, October 2020
39. Moura, G.C.M., Heidemann, J., Hardaker, W., Bulten, J., Ceron, J., Hesselman, C.: Old but gold: prospecting TCP to engineer DNS anycast (extended). Technical report ISI-TR-739, USC/Information Sciences Institute, June 2020. https://www.isi.edu/. Accessed April 2021

40. Moura, G.C.M., Heidemann, J., de O. Schmidt, R., Hardaker, W.: Cache me if you can: effects of DNS time-to-live (extended). In: Proceedings of the ACM Internet Measurement Conference. p. to appear. ACM, Amsterdam, the Netherlands, October 2019
41. Moura, G.C.M., Müller, M., Davids, M., Wullink, M., Hesselman, C.: Fragmentation, truncation, and timeouts: are large DNS messages falling to bits? In: Hohlfeld, O., Lutu, A., Levin, D. (eds.) PAM 2021. LNCS, vol. 12671, pp. 460–477. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72582-2_27
42. Moura, G.C.M., et al.: Anycast vs. DDoS: evaluating the November 2015 root DNS event. In: Proceedings of the ACM Internet Measurement Conference, pp. 255–270. ACM, Santa Monica, November 2016
43. Müller, M., Moura, G.C.M., de O. Schmidt, R., Heidemann, J.: Recursives in the wild: engineering authoritative DNS servers. In: Proceedings of the ACM Internet Measurement Conference, pp. 489–495. ACM, London (2017)
44. Müller, M., Moura, G.C.M., de O. Schmidt, R., Heidemann, J.: Recursives in the wild: engineering authoritative DNS servers. Technical report ISI-TR-720, USC/Information Sciences Institute, September 2017
45. Nanda, P., Simmonds, A.: A scalable architecture supporting qos guarantees using traffic engineering and policy based routing in the internet. Int. J. Commun. Netw. Syst. Sci. (2009)
46. Narten, T., Draves, R., Krishnan, S.: Privacy extensions for stateless address autoconfiguration in IPv6. RFC 4941, IETF, September 2007
47. Partridge, C., Mendez, T., Milliken, W.: Host anycasting service. RFC 1546, IETF, November 1993
48. Pujol, E., Poese, I., Zerwas, J., Smaragdakis, G., Feldmann, A.: Steering hypergiants' traffic at scale. In: Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies, pp. 82–95 (2019)
49. Quoitin, B., Pelsser, C., Bonaventure, O., Uhlig, S.: A performance evaluation of BGP-based traffic engineering. Int. J. Netw. Manage. **15**(3), 177–191 (2005)
50. RIPE NCC: RIPE Atlas measurement IDS, December 2019. https://atlas.ripe.net/measurements/ID. Where ID is the experiment ID: kroot-udp: 10311, kroot-tcp: 26995926, lroot-udp: 10308, lroot-tcp:26995949, GoDNS-21: 23859473, GoTrace-21: 23859475 GoDNS-22: 23863904, GoTrace-22: 23863901, Comcast V6: 24269572, ComcastV6-afterReport: 24867517, ChinaNetV6: 24257938, DNS/TCP:22034303, DNS/UDP: 22034324
51. RIPE NCC: RIPE atlas probes, May 2020. https://ftp.ripe.net/ripe/atlas/probes/archive/2020/05/
52. RIPE Ncc Staff: RIPE atlas: a global internet measurement network. Internet Protocol J. (IPJ) **18**(3), 2–26 (2015)
53. RIPE Network Coordination Centre: RIPE Atlas (2015). https://atlas.ripe.net
54. RIPE Network Coordination Centre: RIPE - Routing Information Service (RIS) (2020). https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
55. Root Server Operators: Events of 2015–11-30, November 2015. http://root-servers.org/news/events-of-20151130.txt
56. Root Server Operators: Root DNS, November 2019. http://root-servers.org/
57. Schlinker, B., Cunha, I., Chiu, Y.C., Sundaresan, S., Katz-Bassett, E.: Internet performance from Facebook's edge. In: Proceedings of the Internet Measurement Conference, IMC 2019, pp. 179–194. ACM, New York (2019)

58. Schlinker, B., et al.: Engineering egress with edge fabric: steering oceans of content to the world. In: Proceedings of the ACM SIGCOMM Conference, pp. 418–431. ACM, Los Angeles, August 2017
59. de Oliveira Schmidt, R., Heidemann, J., Kuipers, J.H.: Anycast latency: how many sites are enough? In: Kaafar, M.A., Uhlig, S., Amann, J. (eds.) PAM 2017. LNCS, vol. 10176, pp. 188–200. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54328-4_14, https://www.isi.edu/
60. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
61. SIDN Labs: ENTRADA - DNS Big Data Analytics, January 2020. https://entrada.sidnlabs.nl/
62. SIDN Labs: .nl stats and data, August 2020. http://stats.sidnlabs.nl
63. Singla, A., Chandrasekaran, B., Godfrey, P.B., Maggs, B.: The internet at the speed of light. In: Proceedings of ACM Hotnets. ACM, Los Angeles, October 2014. http://conferences.sigcomm.org/hotnets/2014/papers/hotnets-XIII-final111.pdf
64. Sommese, R., et al.: Manycast2: using anycast to measure anycast. In: Proceedings of the ACM Internet Measurement Conference, IMC 2020, pp. 456–463. Association for Computing Machinery, New York (2020)
65. Souders, S.: High-performance web sites. Commun. ACM **51**(12), 36–41 (2008)
66. Vixie, P.: Response rate limiting in the domain name system (DNS RRL), June 2012. http://www.redbarn.org/dns/ratelimits, http://www.redbarn.org/dns/ratelimits
67. Wabeke, T., Moura, G.C.M., Franken, N., Hesselman, C.: Counterfighting counterfeit: detecting and taking down fraudulent webshops at a ccTLD. In: Sperotto, A., Dainotti, A., Stiller, B. (eds.) PAM 2020. LNCS, vol. 12048, pp. 158–174. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44081-7_10
68. Wei, L., Flores, M., Bedi, H., Heidemann, J.: Bidirectional anycast/unicast probing (BAUP): optimizing CDN anycast. In: Proceedings of the IEEE Network Traffic Monitoring and Analysis Conference, IFIP, Berlin, June 2020. https://www.isi.edu/
69. Wessels, D.: RSSAC002-data, May 2020. https://github.com/rssac-caucus/RSSAC002-data/
70. Wullink, M., Moura, G.C., Müller, M., Hesselman, C.: ENTRADA: a high-performance network traffic data streaming warehouse. In: 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 913–918. IEEE, April 2016
71. Yu, Y., Wessels, D., Larson, M., Zhang, L.: Authority server selection in DNS caching resolvers. SIGCOMM Comput. Commun. Rev. **42**(2), 80–86 (2012)
72. Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A., Somaiya, N.: Connection-oriented DNS to improve privacy and security. In: Proceedings of the 36th IEEE Symposium on Security and Privacy, pp. 171–186. IEEE, San Jose, May 2015. https://www.isi.edu/
73. Zhu, P., et al.: Characterizing transnational internet performance and the great bottleneck of China. Proc. ACM Meas. Anal. Comput. Syst. **4**(1), 1–23 (2020). https://doi.org/10.1145/3379479