Anycast in Context: A Tale of Two Systems

Thomas Koch
Columbia University

Ke Li
Columbia University

Calvin Ardi
USC/ISI

Ethan Katz-Bassett
Columbia University

Matt Calder
Microsoft/Columbia University

John Heidemann
USC/ISI

ABSTRACT

Anycast is used to serve content including web pages and DNS, and anycast deployments are growing. However, prior work examining root DNS suggests anycast deployments incur significant inflation, with users often routed to suboptimal sites. We reassess anycast performance, first extending prior analysis on inflation in the root DNS. We show that inflation is very common in root DNS, affecting more than 95% of users. However, we then show root DNS latency *hardly matters* to users because caching is so effective. These findings lead us to question: is inflation inherent to anycast, or can inflation be limited when it matters? To answer this question, we consider Microsoft's anycast CDN serving latency-sensitive content. Here, latency matters orders of magnitude more than for root DNS. Perhaps because of this need, only 35% of CDN users experience any inflation, and the amount they experience is smaller than for root DNS. We show that CDN anycast latency has little inflation due to extensive peering and engineering. These results suggest prior claims of anycast inefficiency reflect experiments on a single application rather than anycast's technical potential, and they demonstrate the importance of context when measuring system performance.

CCS CONCEPTS

• **Networks → Network performance analysis**.

KEYWORDS

Anycast, root DNS, routing, latency, CDN.

ACM Reference Format:
Thomas Koch, Ke Li, Calvin Ardi, Ethan Katz-Bassett, Matt Calder, and John Heidemann. 2021. Anycast in Context: A Tale of Two Systems. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3452296.3472891

1 INTRODUCTION

IP anycast is an approach to routing in which geographically diverse servers known as anycast sites all use the same IP address. It is used by a number of operational Domain Name System (DNS) [1, 7, 31, 39, 65] and Content Delivery Network (CDN) [16, 21, 30, 65, 75] deployments today, in part because of its ability to improve latency to clients and decrease load on each anycast server [45, 55, 64].

However, studies have argued that anycast often provides suboptimal performance compared to the lowest latency one could achieve given deployed sites [51, 54, 67]. Notably, the SIGCOMM 2018 paper "Internet Anycast: Performance, Problems, & Potential" has drawn attention to the fact that anycast can inflate latency by hundreds of milliseconds [51], leaving readers of the paper with a poor impression of anycast. Conversely, other work has shown inflation is quite low in Microsoft's anycast CDN [16] and Google Public DNS [50], but used different coverage, metrics, and methodology, so it is difficult to directly compare results. Perhaps because of the very different takeaways of these studies, we have found that some experts in the community have negative opinions of anycast. In particular, it seems surprising that anycast continues to see more adoption and growth in production systems – why continue to use anycast if it causes inflation?

To understand the impact of anycast inefficiency, and its wide use in spite of inflation, we step back and evaluate anycast as a component of actual applications/services. User-affecting performance depends on the anycast deployment, how anycast is used within the service, and how users interact with the service. To see these effects, we consider anycast's role within two real-world systems: the root DNS and Microsoft's anycast CDN serving web content. These applications have distinct goals, they are key components of the Internet, and they are two of the dominant, most studied anycast use cases.

We analyze root DNS [39] packet traces which are available via DITL [26] and which are featured in existing anycast studies [23, 51, 54, 58, 69], with increased coverage compared to prior work. The 13 root letters operate independently with diverse deployment strategies, enabling the study of different anycast deployments providing the same service. We analyze two days of unsampled packet captures from nearly all root DNS letters, consisting of tens of billions of queries from millions of recursive resolvers querying on behalf of all users worldwide, giving us broad coverage.

We also examine Microsoft's CDN using the same methodology we use for the root DNS so we can directly compare results. Microsoft's CDN configures subsets of sites into multiple anycast "rings" of different sizes, providing deployment diversity, but all operated by one organization. We analyze global measurements from over a billion Microsoft users in hundreds of countries/regions, giving us a complete view of CDN performance.

With these measurements, we present the largest study of anycast latency and inflation to date. We first validate and extend prior work on inflation in anycast deployments [51]. Whereas that work focused primarily on a single root letter, we analyze almost the

SIGCOMM '21, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8383-7/21/08…$15.00
https://doi.org/10.1145/3452296.3472891

whole root DNS. By joining root DNS captures with global-scale traces of user behavior, we find that more users than previously thought experience *some* inflation (on average, more than 95%), and as many as 40% of users experience more than 100 ms of inflation to some root letters (§3). However, average inflation per query to the roots is lower than previously thought, since each recursive resolver can preferentially query its best performing root letter – on average, only 10% of users experience more than 100 ms of inflation.

Do recursives have to implement preferential querying strategies for their users so that inflation does not hurt user performance? The answer is a resounding "no" – using new methodology that amortizes DNS queries over users who benefit from cached query results, we find differences in latency and inflation among root letters are *hardly perceived* by users – most users interact with the root DNS once per day (§4). Delay is minimal due to caching of root DNS records with long TTLs at recursive resolvers.

The inflated anycast routes to root DNS could be a result of latency not mattering, causing root operators to not optimize for it, or inflation could be inherent in anycast routing as suggested in prior work. To determine which is the case, we use measurements from Microsoft's CDN and find that, were latency to Microsoft's CDN to be *hypothetically* inflated as to individual root letters, it would result in *hundreds of milliseconds* of additional latency per page load. This increased latency would negatively affect the user's overall experience, especially when compared to root DNS. The key difference is that users incur several RTTs to Microsoft's CDN when fetching web content, whereas users rarely wait for a query to the root DNS because of DNS caching (§5.1).

With this context, we then measure *actual* inflation in Microsoft's CDN and find that inflation is kept comparatively small (§5.2), especially compared to individual root letters. To explain why inflation is so different in these deployments, we contrast AS-level connectivity and inflation between the users, Microsoft's CDN, and roots. We find that Microsoft is able to control inflation through extensive peering and engineering investment (§7.1), even though inefficiency increases with larger deployments (§7.2). Through discussions with operators of root DNS and CDNs, we find recent root DNS expansion has (surprisingly) been driven by a desire to reduce latency and mitigate DDoS attacks, while CDN expansion is driven by market forces (§7.3).

The comparison between performance in these two deployments allows us to put results from prior work in perspective [16, 23, 51, 69]. Even though root inflation is large, users rarely experience it, making its impact on the average query quite small. In contrast, users frequently interact with the CDN, and inflation there is small. These inflation results make sense, given the economic incentives of the organizations running Microsoft's CDN and the root DNS. While we expect these results to hold for other latency-sensitive services using anycast, as they have similar economic incentives, a key takeaway from our work is that anycast must be analyzed in the context of the service in which it is used (§7.3), and so we cannot make definitive statements about generalizability. Hence, we do not refute past claims that anycast can inflate latencies, but we expand on these studies to show that, where it counts, anycast performance can be quite good.

This paper poses no ethical issues.

2 METHODOLOGY AND DATASETS

We use a combination of DNS packet captures and global CDN measurements to measure latency and inflation. Root DNS data is readily available [26], while CDN data is proprietary. We supplement these datasets with measurements from RIPE Atlas [71]. We summarize our many data sets' characteristics, strengths, and weaknesses in Appendix A.

2.1 Root DNS

The first of the two systems we discuss, the root DNS, is a critical part of the global DNS infrastructure. DNS is a fundamental lookup service for the Internet, typically mapping hostnames to IP addresses [22, 56]. To resolve a name to its result, a user sends DNS requests to a recursive resolver (recursive). The recursive queries authoritative DNS servers as it walks the DNS tree from root, to top-level domain (TLD), and down the tree. Recursives cache results to answer future requests according to TTLs of records. The root DNS server is provided by 13 letters [39], each with a different anycast deployment with 6 to 254 anycast sites (as of July 2021), run by 12 organizations. A root DNS site can be local or global – local sites serve small geographic areas or certain ASes (controlled by restricting the propagation of the anycast BGP announcement from the site), while global sites are globally reachable.

We use three datasets: for end-users, we use long-term packet captures from the Information Sciences Institute (ISI) at USC, and DNS and browser measurements from daily use of two of the authors. For DNS servers, we use 48-hour packet captures at most root servers from Day in the Life of the Internet (DITL) [26].

Packet captures from ISI provide a local view of root DNS queries. The recursive resolver runs BIND v9.11.14. The captures, from 2014 to the present, reflect all traffic (incoming and outgoing) traversing port 53 of the recursive resolver. We use traces from 2018 (about 100 million queries), as they overlap temporally with our other datasets. This recursive resolver received queries from hundreds of users on laptops, and a number of desktop and rack-mounted computers of a network research group, so the results may deviate from a typical population. We found no measurement experiments or other obvious anomalies in the period we use.

We use the 2018 DITL captures, archived by DNS-OARC [26], to obtain a global view of root DNS use. DITL occurs annually, with each event including data from most root servers. The 2018 DITL took place 2018/04/10-12 and included 12 root letters (all except G root). Traces from I root are fully anonymized, so we did not use them. Traces from B root are partially anonymized, but only at the /24 level. Our analysis does not rely on addresses more specific than /24, so we use all data from B root and all other roots except G and I. Although the 2018 DITL is older than the most recently available, it is significantly more complete than recent DITLs; in Appendix B.3 we conduct analysis on the 2020 DITL and find none of our main conclusions change.

Since we aim to understand in part how root DNS latency affects users, we filter queries in DITL that do not affect user latency and queries generated by recursives about which we have no user data. We describe this pre-processing of DITL and subsequent joining of root query volumes with Microsoft's CDN user population counts.

Of the 51.9 billion daily queries to all roots, we discard 31 billion queries to non-existing domain names and 2 billion PTR queries. About 28% of non-existing domain name queries are NXDomain hijacking detection from Chromium-based browsers [4, 34, 73], and so involve machine startup and not browsing latency. Prior work suggests the remainder are generated by other malfunctioning, automated software [28]. Similarly, while PTR queries have some uses (traceroutes and confirming hostnames during authentication), they are not part of typical user web latency. In Appendix B.1, we find that including invalid TLD queries significantly changes the conclusions we can draw about how users interact with the root DNS, and we provide more justification for this step. We next remove queries from prefixes in private IP space [38] (7% of all queries). Finally, we analyze only IPv4 data and exclude IPv6 traffic (12% of queries) because we lack v6 user data.

Sources of DNS queries in DITL are typically recursive resolvers, so the captures alone provide no information about how many DNS queries each user makes. To estimate per-user latency, we augment these traces with the approximate number of Microsoft users of each recursive, gathered in 2019 (the oldest user data we have). This user data is from Microsoft DNS data, which counts unique IP addresses as "users". This definition undercounts multiple human users that use a single IP address with Network Address Translation. Microsoft maps recursives to user IP addresses with an existing technique that instruments users to request DNS records for domains Microsoft controls when users fetch content [17, 53].

We join the DITL captures and Microsoft user counts by the recursive resolver /24, aggregating DITL query volumes and Microsoft user IP counts, each grouped by /24 prefix[1] to increase the amount of recursives for which we have user data. This aggregation is justified since many organizations use colocated servers within the same /24 as recursives [31, 63]. Prior work has also found that up to 80% of /24's are collocated [29]. We provide additional justification for this preprocessing step in Appendix B.2, by showing all addresses in a /24 in DITL are almost always routed to the same anycast site. For simplicity, we henceforth refer to these /24's as recursives, even though each /24 may contain several recursives. We call this joined dataset of query volumes and user counts by recursive DITL∩CDN.

In an effort to make our results more reproducible, and as a point of comparison, we also use public Internet population user count data from APNIC to amortize root DNS queries [37] (*i.e.,* instead of using proprietary Microsoft data). APNIC obtains these AS user population estimates by first gathering lists of IP addresses from Google's Ad delivery network, separated by country. APNIC converts this distribution of IP addresses to a distribution of ASNs, normalized by country Internet-user populations. We use the TeamCymru IP to ASN mapping to map IP addresses seen in the DITL captures to their respective ASes [25] and accumulate queries by ASN. We were able to map 99.4% of DITL IP addresses to an ASN, representing 98.6% of DITL query volume. The assumption that recursives are in the same AS as the users they serve is obviously incorrect for public DNS services, but we do not make an effort to correct for these cases. Overall, we believe Microsoft user counts are more accurate, but APNIC data is more accessible to other researchers and so provides a useful comparison.

[1]We aggregate user IP addresses by recursive /24 before counting to ensure we do not double-count users.

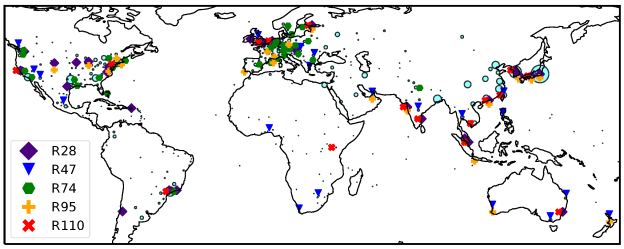


Figure 1: Microsoft's CDN rings and user populations. Sites in smaller rings are also in larger rings, and the legend indicates the number of sites in that ring. We do not show some front-ends too close to each other to improve readability. User populations are shown as circles, with the radius of the circle proportional to the number of users in that region, demonstrating that Microsoft has deployed front-ends in areas of user concentration.

2.2 Microsoft's CDN

We also analyze Microsoft's large anycast CDN that serves web content to over a billion users from more than 100 sites. Traffic destined for Microsoft's CDN enters its network at a point of presence (PoP) and is routed to one of the anycast sites serving the content (front-ends). Microsoft organizes its deployment into groups of sites, called rings, that conform to varying degrees of regulatory restrictions (*e.g.,* ISO 9001, HIPAA), each with its own anycast address. The rings have the property that a site in a smaller ring is also in all larger rings. Other CDNs have to work with similar regulatory restrictions [2]. Hence, traffic from a user prefix destined for Microsoft's CDN may end up at different front-ends (depending on which ring the application uses), but often will ingress into the network at the same PoP. Users are routed to rings via anycast and fetch web content from a front-end via its anycast address. Users are always routed to the largest *allowed* ring given the application's regulatory restrictions (performance differences among rings are not taken into account).

Microsoft's anycast rings provide different size anycast deployments for study. In Figure 1 we show Microsoft's front-ends and user concentrations. Rings are named according to the number of front-ends they contain, and front-ends are labeled according to the smallest ring to which they belong (or else all front-ends would be labelled as R110). We do not show some front-ends too close to each other to improve readability. Circles are average user locations, where the radius of the circle is proportional to the population of users in that region. Figure 1 suggests that front-end locations tend to be near large populations, providing at least one low latency option to most users. Appendix F illustrates latency differences by region.

User locations are aggregated by *region*, a geographic area used internally by Microsoft to break the world into *regions* that generate similar amounts of traffic and so contain similar numbers of users. A region often corresponds to a large metropolitan area. We refer to users at the ⟨`region, AS`⟩ granularity, because users in the same ⟨`region, AS`⟩ location are often routed to the same front-ends and so (generally) experience similar latency. There are 508 regions in

total: 135 in Europe, 62 in Africa, 102 in Asia, 2 in Antarctica, 137 in North America, 41 in South America, and 29 in Oceania.

To study performance in Microsoft's CDN, we use two major data sources: server-side logs and client-side measurements. Server-side logs at front-ends collect information about user TCP connections, including the user IP address and TCP handshake RTT. Using these RTTs as latency measurements, we compute median latencies from users in a ⟨region, AS⟩ location to each front-end that serves them.[2] Microsoft determines the location and AS of users using internal databases.

Client-side measurements come from a measurement system operated by Microsoft [17]. Latency measurements are the time it takes for Microsoft users to fetch a small image via HTTP.[3] The measurement system instructs clients using CDN services to issue measurements to multiple rings, which enables us to remove biases in latency patterns due to services hosted on different rings having different client footprints (*e.g.*, enterprise versus residential traffic). Microsoft collects latencies of users populations, noting the location and AS of the user. Since these measurements come directly from end-users, we do not know which front-end the user hit. For both client-side measurements and server-side logs, we collect statistics for over a billion users across 15,000 ⟨region, AS⟩ locations.

We also use RIPE Atlas to ping anycast rings, because we cannot share absolute latency numbers. We calibrate these results versus our (private) data measuring latency for CDN users. In total, we collect 7,000 ping measurements to rings from 1,000 RIPE Atlas probes in more than 500 ASes to augment CDN latency measurements. (Probes were selected randomly, and measured three times to each ring.)

3 ROUTES TO ROOT DNS ARE INFLATED

Earlier work has found query distance to the root DNS is often significantly inflated [13, 23, 51, 67, 69]. Similar to this work, we find that queries often travel to distant sites despite the presence of a geographically closer site. We extend this understanding in a number of ways. While previous work considered only subsets of root DNS activity and focused on geographic inflation for recursives rather than users, we calculate inflation for nearly all root letters, and place inflation in the context of *users*, rather than recursive resolvers. These contributions are significant for several reasons. First, considering more root letters allows us to evaluate inflation in different deployments, and with most letters we can evaluate the root DNS *system*. Since a recursive makes queries to many root letters, favoring those with low latency [60], *system* performance and inflation can (and does) differ from component performance. Second, we weight recursive resolvers by the number of users, which allows us to see how users are affected by inflation. Finally, we extend prior work by conducting an analysis of latency (as opposed to geographic) inflation with large coverage.

Previous studies of anycast have separated inflation into two types, unicast and anycast, in an attempt to tease out how much latency anycast specifically adds to queries [13, 16, 51, 69]. For several reasons, we choose to consider inflation relative to the deployment, rather than try to infer which inflation would exist in an equivalent unicast deployment. First, coverage of measurement platforms used to determine unicast inflation such as RIPE Atlas (vantage points for anycast studies [51, 69]) is not representative [10]. Second, calculating unicast inflation requires knowledge of the best unicast alternative from every recursive seen in DITL to every root letter, something that would be difficult to approximate with RIPE Atlas because some letters do not publish their unicast addresses. Third, we find it valuable to compare latency to a theoretical lower bound, since user routes to the best unicast alternative may still be inflated.

We measure two types of inflation for the root DNS, by looking at which sites recursive resolvers are directed to. DITL captures are a rich source of data because they provide us with a global view of which recursives access which locations (§2.1). Our inflation analysis covers 224 countries/regions and 22,243 ASes (Atlas covers about 3,700 ASes as of July 2021).

We calculate the first type of inflation – geographic inflation (Eq. (1)) – over 10 of the 13 root letters, omitting G which does not provide data, H which only had one site in 2018 (and so has zero inflation), and I, where anonymization prevents analysis. Geographic inflation measures, at a high level, how users are routed to sites compared to the closest front-end (*i.e.*, efficiency)[4].

We calculate the second type of inflation – latency inflation (Eq. (2)) – over the root letters mentioned above by looking at the subset of DNS queries that use TCP, using the handshake to capture RTT [57]. Our latency inflation analysis further excludes D and L root, due to malformed DITL PCAPs. Latency inflation uses measured latencies to determine inflation, so it reflects constraints due to physical rights-of-way and connectivity, bad routing, and peering choices. We calculate median latency over each ⟨root, resolver /24, anycast site⟩ for which we have at least 10 measurements, providing us latencies for resolvers representing 40% of DITL query volume to these roots.

3.1 Methodology

To calculate geographic inflation, we first geolocate all recursives in our DITL∩CDN dataset using MaxMind [41], following prior methodology which affirmed MaxMind to be suitably accurate for geolocating recursive resolvers in order to assess inflation [51]. We then compute geographic inflation (scaled by the speed of light in fiber) for each recursive sending queries to root server j as

$$GI(R, j) = \frac{2}{c_f}\left(\sum_i \frac{N(R, j_i)d(R, j_i)}{N(R, j)} - \min_k d(R, j_k)\right) \quad (1)$$

where $N(R, j_i)$ is the number of queries to site j_i by recursive R, $N(R, j) = \sum_i N(R, j_i)$ is the total number of queries to all sites j_i in root j by recursive R, c_f is the speed of light in fiber, the factor of 2 accounts for the round trip latency, $d(R, j_k)$ is the distance between the recursive resolver and site j_k, and both the summation and minimization are over the global sites in this letter deployment (see Section 2.1 for the distinction between local and global). We only consider global sites, since we do not know which recursives can reach local sites. For recursives which can reach a local site

[2] We also looked at other percentiles (*e.g.*, 95th) and found the qualitative results to be similar.

[3] DNS resolution and TCP connection time are factored out.

[4] It would be interesting to measure topological inflation (extra distance traveled on the Internet topology, beyond shortest-path propagation-delay), but it would be difficult to do so using existing methods without sacrificing significant coverage.

but instead reach a global site, Equation (1) (and Equation (2)) may underestimate actual inflation.

$GI(R, j)$ is an approximation of the inflation one would expect to experience when executing a single query to root deployment j from recursive R, averaged over all sites. The overall geographic inflation of a recursive is then the empirical mean over all roots. Even though queries from the same recursive /24 are usually routed together, they may be routed to different sites due to load balancing in intermediate ASes (see Appendix B.2 for measures of how often this occurs), so we average geographic inflation across sites for a recursive. Geographic inflation is useful to investigate since it shows how our results compare with prior work, how many users are being inflated, and it gives us a measure of "efficiency" (§7.2) .

We also calculate latency inflation, again considering recursive querying patterns seen in DITL. We calculate latency inflation $LI(R, j)$ for users of recursive R to root j as

$$LI(R, j) = \sum_i \frac{N(R, j_i)l(R, j_i)}{N(R, j)} - \frac{3 \times 2}{2c_f} \min_k d(R, j_k) \qquad (2)$$

where $l(R, j_i)$ is the median latency of recursive R towards root site j_i and the other variables are as in Equation (1). Prior work notes that routes rarely achieve a latency of less than the great circle distance between the endpoints divided by $\frac{2c_f}{3}$ [46], so we use $\frac{2c_f}{3}$ to lower bound the best latency recursives could achieve. Latency inflation is a measure of potential performance improvement users could see due to changes in routing or expanding the physical Internet (*e.g.,* laying fiber).

One limitation is that we do not account for the fact that the source addresses of some queries in the DITL traces may be spoofed. Spoofing is more likely to make our calculated inflation larger, especially in cases where the spoofer is far away from the physical interface it is spoofing (*i.e.,* from our perspective, the route looks inflated when actually the source address was spoofed). We do not attempt to correct for these cases since it would be difficult to distinguish between legitimately poor routing and spoofed traffic.

3.2 Results

Figure 2a demonstrates that the likelihood of a root DNS query experiencing any geographic inflation (Eq. (1)) roughly grows with deployment size (y-axis intercept), expanding on results in prior work which presented an orthogonal, aggregated view [51]. The `All Roots` line takes into account that each recursive spreads its queries across different roots. It has the lowest y-intercept of any line in Figure 2a, which implies that nearly every recursive experiences some inflation to at least one root and that the set of inflated recursives varies across roots. Hence, our analysis shows that nearly every user will (on average) experience inflation when querying the root DNS, and 10.8% of users are likely to be inflated by more than 2,000 km (20 ms).

Figure 2b shows that queries to these roots experience frequent latency inflation (Eq. (2)), with between 20% and 40% of users experiencing greater than 100 ms of inflation (B root is a clear exception, but only had 2 sites, so inflation is less meaningful). Latency inflation starts at approximately zero, which follows from our choice of "optimal" latency (Eq. (2)). Compared to geographic inflation, latency inflation is particularly larger in the tail. For example, at

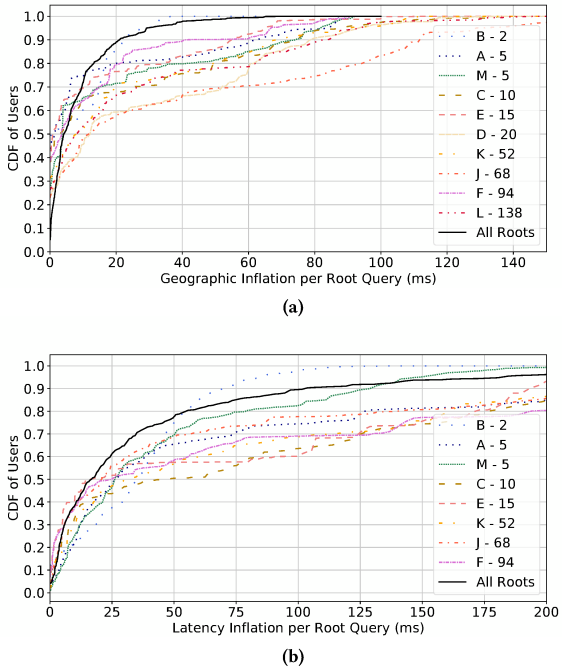


Figure 2: Inflation measured using geographic information (2a) and TCP RTT estimates (2b). Generally, larger deployments are more likely to inflate paths, and inflation in the roots is quite large. The legends indicate the number of global sites per letter during the 2018 DITL.

the 95^{th} percentile `C` root has 240 ms of latency inflation but only 70 ms of geographic inflation. However, inflation for *the root DNS* as a whole is not as bad as individual root letters as shown by lines `All Roots`, which take into account that recursives can preferentially query low latency root servers [60].

Our latency inflation metric shows `C` root is more inflated than previously thought, inflating 35% of users by more than 100 ms compared to 20% reported in prior work [51] (although the comparison to prior work is not perfect since what was measure is different). Other prior work found significant inflation in the roots, but it is difficult to directly compare results since inflation was presented in different ways [23, 69].

Clearly, routing to individual root letters often is inflated, with many queries traveling thousands more kilometers than needed, and being inflated by hundreds of milliseconds for some users.

4 ROOT DNS LATENCY AND INFLATION HARDLY MATTER

With a richer understanding of inflation in the root DNS, one might wonder why inflation in root letters is large given growing deployments and root DNS's importance in the Internet. We now show that root DNS inflation does not result in much *user-visible* latency.

4.1 Measuring Root DNS Latency Matters

The root DNS servers host records for TLDs (*e.g.,* COM, ORG). There are approximately one thousand TLDs, and nearly all of the

corresponding DNS records have a TTL of two days. Hence, due to shared caches at local resolvers, one might think root DNS latency *trivially* does not matter for users. Recent work even suggests the root DNS can be done away with entirely [5] or largely replaced by preemptive caching in recursives [48]. We offer several reasons why we found it necessary to explicitly measure root DNS latency's impact on users, rather than use intuition.

First, there is a lot of attention being placed on the root DNS in the professional and research communities. For example, some experts have asked us in conversation why CDNs use anycast, when anycast inflates latencies in the root DNS so much. The SIGCOMM 2018 paper "Internet Anycast: Performance, Problems, & Potential" has drawn attention to the fact that anycast can inflate latency to the root DNS by hundreds of milliseconds [51]. Blog posts from the root letters discuss latency improvements and inflation reductions [3, 14, 61, 79] – why does latency matter to roots? Moreover, over the past 5 years the number of root DNS sites has steadily increased to more than double, from 516 to 1367. Why is there so much investment in more sites?

Second, there is value in quantitatively analyzing systems, especially global systems that operate at scale, even if we can intuitively, qualitatively reason about these systems without conducting analysis. We conduct analysis using data from eleven of thirteen root letters, giving us a truly global view of how users interact with the root DNS. We are aware of only one other study which looked at how caching affects root DNS queries [44], but that study is old, is limited to one recursive resolver, and does not place DNS queries in the context of user experience.

Third, although TTLs of TLD records are two days, recursive resolver implementations can be buggy. We noticed millions of queries per day for TLD records being sent to the root letters by some recursives (§4.3), and found a bug in the popular BIND recursive resolver software that causes unnecessary queries to the roots (Appendix E). Hence, making arguments about root DNS latency requires careful analysis.

4.2 How We Measure Root DNS

Measuring how root DNS latency affects users poses several challenges. To put root DNS latency into context we must understand (1) how user-application performance is affected when applications make root queries, (2) how often end-hosts and recursive resolvers interact with root DNS, given their caches, (3) what the latency is from the anycast deployment, and (4) how these effects vary by location and root letter. These challenges both motivate our subsequent analyses and also highlight the limitations of prior work which do not capture these subtleties of root DNS latency [23, 51, 58, 69].

Therefore, precisely determining how root DNS latency affects users would require global, OS-level control to select recursives and view OS DNS caches; global application-level data to see when DNS queries are made and how this latency affects application-performance; global recursive data to see caches, root queries, and their latencies; and global root traces to see how queries to the roots are routed. As of July 2021, only Google might have this data, and assembling it would be daunting.

To overcome these challenges we take two perspectives of root DNS interactions: local (close to the user) and global (across more than a billion users). Our local perspective precisely measures how root DNS queries are amortized over users browsing sessions, while our global analysis estimates the number of queries users worldwide execute to the roots.

4.3 Root DNS Latency Hardly Matter

Local Perspective: To obtain a precise measure of how root DNS queries are amortized over a small population, we use packet captures of a recursive resolver at ISI (§2.1). We also measure from two authors' computers to observe how an individual user interacts with the root servers (with no shared cache), since ISI traces do not give us context about user experience. Data from two users is limited, which is a reflection of the challenges we identified in Section 4.2. However, these experiments offer *precise* measures of how these authors interact with root DNS (which no prior work has investigated), supplementing the global-scale data used for most of the paper.

Using traces gathered at ISI, we calculate the number of queries to any root server as a fraction of user requests to the recursive resolver. We call this metric the root cache miss rate, as it approximates how often a TLD record is not found in the cache of the recursive in the event of a user query. It is approximate because the resolver may have sent multiple root requests per user query, and some root requests may not be triggered by a user query. The daily root cache miss rates of the resolver range from 0.1% to 2.5% (not shown), with a median value of 0.5%. The overall cache miss rate across 2018 was also 0.5%. The particular cache miss rate may vary depending on user querying behavior and recursive resolver software, but clearly the miss rate is small, due to shared caches. Appendix D shows the minimal impact root DNS latency has on users of ISI and a CDF of DNS latency experienced by users at ISI.

Since the measurements at ISI can only tell us how often root DNS queries are generated, we next look at how root DNS latency compares to end-user application latency. On two authors' work computers (in separate locations), we direct all DNS traffic to local, non-forwarding, caching recursive resolvers running BIND 9.16.5 and capture all DNS traffic between the user and the resolver, and between the resolver and the Internet.

We run the experiment for four weeks and observe a median daily root cache miss rate of 1.5% – similar to but larger than the cache miss rate at ISI. The larger cache miss rate makes sense, given the local users do not benefit from shared caches. We also use browser plugins to measure median daily active browsing time and median daily cumulative page load time, so we can place DNS latency into perspective. Active browsing time is defined as the amount of time a user spends interacting with the page (with a 30 second timeout), whereas page load time is defined as the time until the window.onLoad event. Median daily root DNS latency is 1.6% of median daily page load time and 0.05% of median daily active browsing time, meaning that root DNS latency is barely perceptible to these users when loading web pages, even without shared caches. In general, we *overestimate* the impact of DNS and root DNS latency since DNS queries can occur as a result of any application running on the authors' machines (not just browsing).

Global Perspective: Towards obtaining a global view of how users interact with the root DNS, we next look at global querying

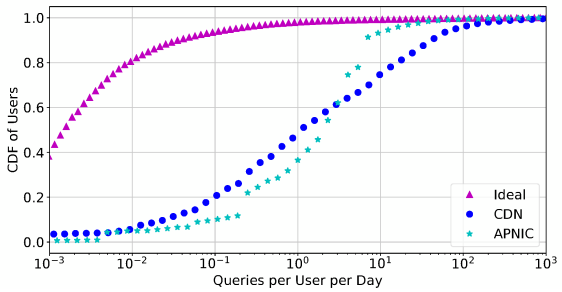


Figure 3: A CDF of the number of queries each user executes to the roots per day. The CDN and APNIC lines represent different user-count datasets. The Ideal line presents an idealized assumption about recursive query behavior. Most users wait for less than one query to the roots per day, regardless of which user data we use.

behavior of recursives. As discussed in Section 4.2, it is difficult to model caching at resolvers and how caching saves users latency, since caching hides user query patterns (by design) and differs with recursive implementation. To overcome this challenge, we use a new methodology that amortizes queries over large user populations, by joining DNS query patterns with user data.

Given query volumes towards root servers from recursives and user counts using each recursive from the DITL captures (§2.1), we estimate the number of queries to the roots that users wait for per day. Figure 3 is a CDF of the expected number of queries per user per day, where lines CDN and APNIC use a different user-count dataset (§2.1), and line Ideal uses hypothetical assumptions which we describe below. Figure 3 demonstrates that most users wait for no more than one query to the roots per day, regardless of which user data we use.

To generate each line in Figure 3, we divide (*i.e.,* amortize) the number of queries to the root servers made by each recursive by the number of users that recursive represents. We weight this quotient (*i.e.,* daily queries per user) by user count and calculate the resulting CDF. We calculate the number of queries per day each recursive makes from DITL by first calculating daily query rates at each site (*i.e.,* total queries divided by total capture time) and subsequently summing these rates across sites. We include nearly every root query captured across the root servers, so Figure 3 provides a truly global view of how users interact with the root DNS.

The two lines CDN and APNIC correspond to amortizing DITL queries over Microsoft and APNIC user counts, respectively. Hence, the set of 'users' each line represents is technically different, but we place them on the same graph for comparison. Even though the two methodologies of estimating user counts behind root queries are very different (CDN uses an internal measurement system, while APNIC uses Internet population estimates by country), amortizing queries over these sets of users still yields the same high level conclusions about how users interact with the root DNS, suggesting that our methodology and conclusions are sound – users *rarely* interact with the root DNS executing about one query per day at the median. Users in the tail are likely either spammers, have buggy recursive software, or represent recursives with more users than DITL∩CDN suggests (*e.g.,* cellular networks). APNIC user estimates are not affected by NATs, and APNIC has a smaller tail.

The line labeled Ideal does not use DITL query volumes to calculate daily user query counts, but instead represents a hypothetical scenario in which each recursive queries for all TLD records exactly once per TTL, and amortizes these queries uniformly over their respective user populations (we use Microsoft user counts for Ideal). The resulting hypothetical median daily query count of 0.007 could represent a future in which caching works at recursives optimally – not querying the roots when not necessary. Ideal also demonstrates the degree to which the assumption that recursives only query once per TTL *underestimates* the latency users experience due to the root DNS (§4.2) – the assumption is orders of magnitude off from reality.

We have shown root DNS latency, and therefore inflated routes to the roots, makes no difference to most users. This result raises the question – are paths to the roots inflated because anycast intrinsically results in inflation? Or rather, does latency not mattering in this setting lead to anycast deployments that are not optimized for latency and hence tend to have inflated routes? To answer these questions, we turn to a new system using anycast to serve latency-sensitive content – Microsoft's CDN.

5 LATENCY MATTERS FOR MICROSOFT'S CDN

We demonstrate that latency (and hence inflation) *does* matter for Microsoft users when fetching web content, unlike for most users in the root DNS, principally due to the number of RTTs users incur when fetching web content.

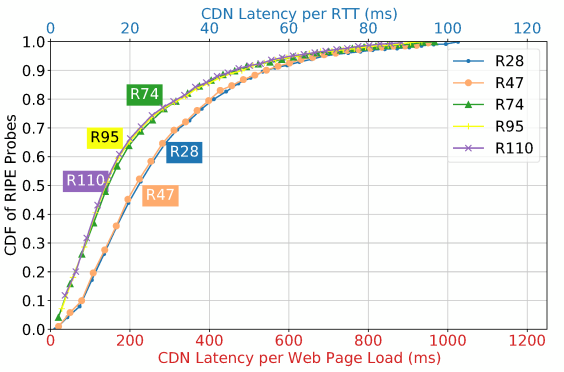
5.1 RTTs in a Page Load

To estimate the latency a user experiences when interacting with Microsoft's CDN (§5.2), we first estimate the number of RTTs required to load a typical web page hosted by Microsoft's CDN.

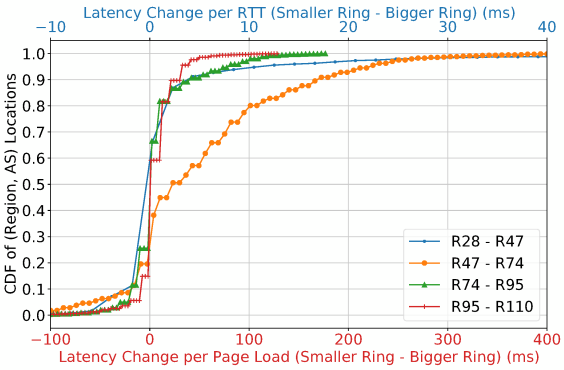
The number of RTTs in a page load depends on a variety of factors, so we aim to lower bound the number. We lower bound the number of RTTs since a lower bound is a conservative measure of the impact of CDN inflation, as the latency inflation accumulates with each additional RTT, and larger pages (more RTTs) would be impacted more. We provide an estimate of this lower bound based on modeling and evaluation of a set of web pages hosted by Microsoft's CDN using Selenium (a headless web browser), finding that 10 RTTs is a reasonable estimate. Due to length restrictions, we include the full details of our measurements and methodology in Appendix C.

5.2 Microsoft's CDN User Latency

We now measure how users are impacted by latency of Microsoft's CDN. First, using measurements from RIPE Atlas probes, we demonstrate that CDN latency results in significant delay to users when fetching web content. Then, using both client-side measurements and server-side logs, we also show that latency usually decreases with more sites. Consequently, Microsoft has a major incentive to limit inflation experienced by users, and investments in more anycast sites positively affect user experience much more in the case of Microsoft's CDN than in the roots. The positive effect on user experience has been a major reason for recent expansion (§7.3).



(a)



(b)

Figure 4: RTTs and latencies per web page load from RIPE probes to CDN rings (4a), and change in median latency for Microsoft users in ⟨region, AS⟩ locations when transitioning rings (4b). Axes with per-RTT latencies are blue, while axes with per-page-load latencies are red. Latencies per page load can be significant, so Microsoft has an incentive to reduce inflation.

Microsoft's CDN has groups of sites called *rings* (§2.2). Each larger ring adds some sites to those of the smaller ring. Each ring provides an IP anycast CDN, so we report results for each of the rings individually. Different ring sizes reflect some of the benefit of additional anycast locations, but a user's traffic usually ingresses to Microsoft's network at the same PoP regardless of ring, since all routers announce all rings. Users experience latency from Microsoft's as they retrieve web objects (*e.g.*, web pages or supporting data) hosted by Microsoft's CDN. Hence, in order to assess how Microsoft users experience latency, we must measure what the RTT is from users to front-ends and how many RTTs are incurred when fetching web content. We use our estimate from Section 5.1 that users incur *at least* 10 RTTs in a page load. To obtain per-page-load latency, we scale anycast latency by the number of RTTs.

In Figure 4a, we show latencies to rings. Figure 4a uses latencies measured from RIPE Atlas probes (§2.2), as we cannot share absolute latencies from Microsoft measurements since Microsoft considers this data proprietary. Although RIPE Atlas has limited coverage [10], we compare (but cannot share) to CDN measurements, which contain latencies from all ⟨region, AS⟩ locations to all rings. We observed that the distribution of RIPE Atlas probe latencies is overall somewhat lower than that of Microsoft's users globally (not shown in figure), so Figure 4a likely underestimates the latency users typically experience.

Users can experience up to 1,000 ms in anycast latency per page load, and, for large deployments (*e.g.*, R95), half of RIPE Atlas probes experience approximately 100 ms of latency per page load (Fig. 4a). Therefore, unsurprisingly, latency to Microsoft's CDN factors into user experience, and so Microsoft has an incentive to decrease latency for users. The difference in median latency per page load between R28 and R110 is approximately 100 ms, which is a measure of how investments in more front-ends can help users. Similarly, a root deployment with more sites tends to have lower latency than a root deployment with fewer sites (§7.2), but such reductions in latency hardly affect user experience (§4).

Latency benefits with more sites are not uniform, and performance falls into one of two "groups" – R28 and R47 have similar aggregate performance, as do R74, R95, and R110. This grouping corresponds to the way rings "cover" users – R74 provides a *significant* additional number of Microsoft users with a geographically close front-end over R47 (§7.2).

To show how adding front-ends tends to help individual ⟨region, AS⟩ locations (in addition to aggregate performance), Figure 4b shows the difference in median latency for a ⟨region, AS⟩ location from one ring to the next larger ring, calculated using CDN measurements (as opposed to RIPE Atlas probes). Most ⟨region, AS⟩ locations experience either equal or better latency to the next largest ring, with diminishing returns as more front-ends are added. A small fraction of users experience small increases in latency when moving to larger rings – 90% of users experience a decrease of at most a few millisecond increase and 99% experience less than a 10 ms increase. Hence, Microsoft does not sacrifice fairness for performance improvements.

We next investigate if Microsoft's clear incentive to reduce latency (and therefore inflation) translates to lower inflation from users to Microsoft's CDN than from users to the root DNS.

6 ANYCAST INFLATION CAN BE SMALL

We next investigate whether Microsoft's incentive to reduce inflation translates to an anycast deployment with less inflation than in the roots, representing the study of anycast CDN inflation with the best coverage to date – measurements are from billions of users in hundreds of countries/regions and 59,000 ASes. Critically, we are able to directly compare inflation between root DNS and Microsoft's CDN, since we use the same methodology with broad coverage.

To measure anycast inflation for Microsoft's CDN we use geographic information and server-side measurements (§2.2). Server-side logs give us a global view of which clients hit which front-ends and the latencies they achieved. Latency is measured via server-side logging of TCP round-trip times. Front-ends act as TCP proxies for fetching un-cached content from data centers. Routing over the global WAN is near optimal [36], so measuring inflation using latency to front-ends (as opposed to measuring inflation using end to end latency) captures all routing inefficiency. We also use Microsoft user locations, which are determined using an internal database.

As in Section 3, we calculate both geographic and latency inflation. We calculate geographic inflation as in Equation (1), except all users in a ⟨region, AS⟩ location are assigned the mean location

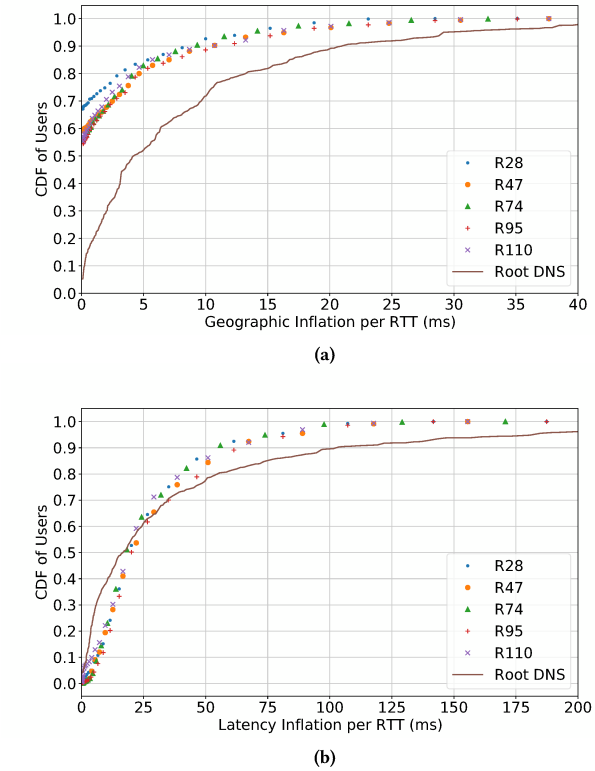


Figure 5: Inflation measured using geographic information (5a) and CDN server side logs (5b). Inflation is more prevalent for larger deployments but is still small for most users.

of users in the ⟨region, AS⟩ location. Anycast inflation results in extra latency for every packet (and corresponding ACK) exchanged between a client and an anycasted service, resulting in a per RTT cost, so we refer to inflation as "per RTT". Application-layer interactions may incur this cost multiple times (as in the case of loading a large web object from a CDN) or a single time (as in the case of typical DNS request/response over UDP).

Microsoft users usually experience no geographic inflation (Fig. 5a, y-axis intercepts), and 85% of users experience less than 10 ms (1,000 *km*) of geographic inflation per RTT for all rings. Conversely, 97% of root DNS users experience some geographic inflation, and 25% of users experience geographic inflation more than 10 ms (1,000 *km*) per RTT. The fact that geographic inflation is larger and more prevalent in the roots than in Microsoft's CDN (at every percentile) suggests Microsoft optimizes its deployment to control it (§7).

We next calculate latency inflation for each ring as in Equation (2). We calculate median latencies over user populations within a ⟨region, AS⟩ location hitting a front-end in a given ring, the assumption being that measurements from some users in a ⟨region, AS⟩ location hitting the same site are representative of all users in that ⟨region, AS⟩ location hitting that site. More than 83% of such medians were taken over more than 500 measurements, so our observations should be robust. There is roughly constant latency inflation as the number of front-ends grows (Fig. 5b), which highlights that even though users have more low latency options (front-ends), they can still take circuitous routes to close front-ends. However, Microsoft is able to keep latency inflation below 30 ms for 70% of users in *all* rings and below 60 ms for 90% of users. In Microsoft's CDN, 99% of users experience less than 100 ms of inflation, but 10% experience more than 100 ms to the roots.

An interesting takeaway from Figure 5b is that system-wide per-query root DNS inflation is quite similar to CDN inflation, a fact that is not clear from prior work [16, 51] since prior work used different methodology and looked at fewer root letters. However, inflation in *individual* root letters is quite worse than in Microsoft's CDN (Fig. 2b). Although inflation in the roots does not matter to most users (§4.3), it is still interesting to see how recursive resolvers can take advantage of the thirteen independent deployments of root letters, and choose which letter is the best for them, in a way that is not possible in Microsoft's CDN.

Compared to prior work which also studied inflation in Microsoft's CDN [16], we find an improvement – 95% of users experience inflation under 80 ms now compared to 85% 5 years ago. This improvement (representing millions of users) is despite the fact that Microsoft's CDN has more than doubled in size and that we use a stricter measure of inflation, and is evidence that expansion reduces efficiency (in terms of % of users at their closest site) but inflation can be kept low through careful deployment (§7.2). Figure 5b also offers a complementary view of inflation compared to prior work [16], which does not take into account that routing from a ⟨region, AS⟩ location to *all* front-ends might be sub-optimal.

Compared to Figure 5a, Figure 5b demonstrates there is room for improvement – at least half of users visit their closest front-end, but those users might take circuitous routes to those front-end as shown by the low y-axis intercepts in Figure 5b. There is still room for latency optimization in anycast deployments, which is an active area of research [43, 47, 82].

7 INCENTIVES AND INVESTMENT SHAPE DEPLOYMENTS AND PATHS

We have definitively answered the questions regarding inflation that we posed at the end of Section 4.3. We now investigate why inflation is so different in root DNS and Microsoft's CDN by looking at path lengths (§7.1), investigate how geographical differences in deployments affect inflation (§7.2), and present reasons behind the expansion of both root DNS and CDNs (§7.3).

7.1 Microsoft's CDN Has Shorter AS Paths, and Short AS Paths are More Direct

CDNs have a financial incentive to keep latency low for users and have the resources to build efficient systems. Microsoft deploys state-of-the-art network routing automation [68, 80], a global SDN WAN [36, 42], and expensive peering agreements when they make economic sense and/or help user experience. These strategies result in short, low latency routes between users and Microsoft.

We can capture some of these engineering efforts by measuring how Microsoft connects to users. CDNs peer widely with end-user networks and so have direct paths to many users [54, 78]. With fewer BGP decision points, paths are often less inflated [70]. This intuition motivates the following investigation of AS path lengths

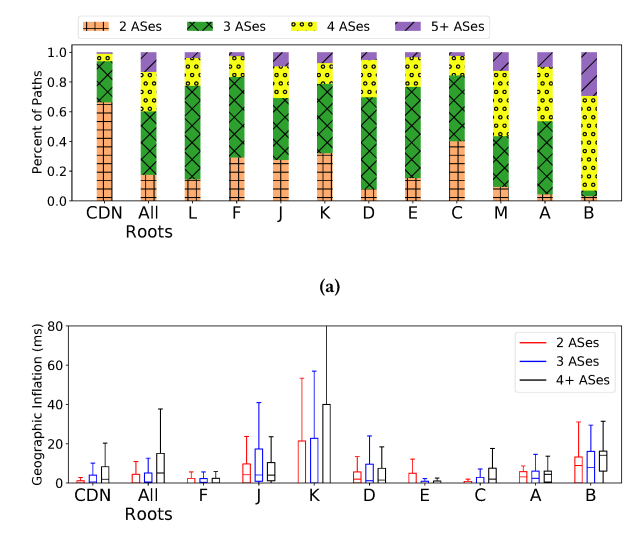


Figure 6: Distribution of the number of ASes traversed to reach various destinations (6a) and the correlation between the AS path length towards a destination and geographic inflation (6b). Microsoft is closely connected to many eyeball ASes, and this connectivity correlates with lower inflation. We group paths towards roots and Microsoft by ⟨`region, AS`⟩ locations, except for 'All Roots' which groups paths by ⟨`region, AS, root`⟩ locations.

towards roots and Microsoft and of how path lengths relate to inflation, which is summarized by Figure 6. Figure 6 quantifies one key difference between root DNS and CDN deployments, but publicly available data cannot capture all of Microsoft's optimizations.

To quantify differences in AS path length between Microsoft and roots, Figure 6a shows AS path lengths to roots and Microsoft from RIPE Atlas probes. We use the maximum number of active RIPE Atlas probes for which we can calculate AS paths to all destinations, amounting to 7,200 RIPE Atlas probes in 158 countries/regions and 2,400 ASes. Although RIPE Atlas probes do not have representative coverage [10], it is the best publicly available system, and we are only interested in qualitative, comparative conclusions.

Lengths towards Microsoft's CDN are based on traceroutes from active Atlas probes in August 2020, whereas lengths towards the roots are based on traceroutes from RIPE Atlas probes in April 2018 (the time of DITL).[5] We perform IP to AS mapping using Team Cymru [25], removing IP addresses that are private, associated with IXPs, or not announced publicly by any ASes. We merge AS siblings together into one 'organization'. We derive sibling data from CAIDA's AS to organization dataset [15]. We group paths by ⟨`region, AS`⟩ location, except for 'All Roots', for which we group paths by ⟨`region, AS, root`⟩ location. We assign each ⟨`region, AS`⟩ location equal weight; when a given ⟨`region, AS`⟩ location hosts multiple RIPE Atlas probes that measure different path lengths to a given destination, the location's weight is split evenly across the measured lengths.

[5]We use AS path lengths from traceroutes towards the roots measured in 2018 in Figure 6, so that we can pair AS path length directly with 2018 DITL inflation data.

Figure 6a shows shorter paths to Microsoft than to the roots. (Weighting by traffic volumes yielded similar results.) 69% of all paths to Microsoft only traverse two ASes (direct from RIPE Atlas probe AS to destination AS), and only 5% of paths to Microsoft traverse four or more ASes. Conversely, between 5% and 44% paths to root letters only traverse two ASes, and between 12% and 63% of paths to roots traverse four or more ASes.

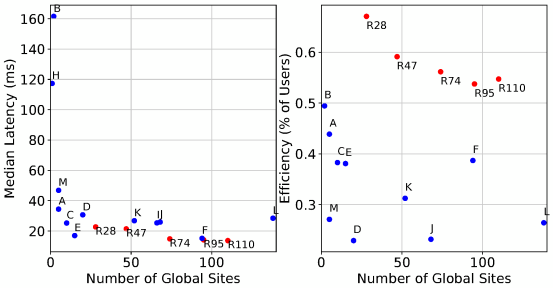
To demonstrate how short AS paths tend to have lower inflation, Figure 6b shows the correlation between AS path length and geographic inflation[6]. We compare to geographic (as opposed to latency) inflation since we are able to calculate it for more root letters. For the inflation towards destinations in Figure 6b, we use the geographic inflation associated with that ⟨`region, AS`⟩ location calculated for Figure 2 and Figure 5a. The AS path length towards each destination is the most common AS path length measured across RIPE Atlas probes in the same ⟨`region, AS`⟩ location. Figure 6b demonstrates that paths that traverse fewer ASes tend to be inflated less. `All Roots` shows that this is true globally, across root letters, and the results for each individual root letter shows geographic inflation is less for paths traversing 2 ASes than it is for paths traversing more (except for B and E root). The relationship between inflation and AS path length is very different across root letters, which is evidence of different deployment strategies.

Overall, our results demonstrate that shorter paths tend to have less inflation, users have shorter paths to Microsoft than towards the roots, and Microsoft tends to have less inflation across path lengths. We believe these observations are a result of strategic business investments that Microsoft puts toward peering and optimizing its routing and infrastructure. In addition to shorter AS paths generally being less inflated [70], direct paths to Microsoft's CDN in particular sidestep the challenges of BGP by aligning the best performing paths with the BGP decision process [20]. Direct paths will usually be preferred according to BGP's top criteria, local preference and AS path length (because by definition they are the shortest and from a peer, and ASes usually set local preference to prefer peer routes in the absence of customer routes, which for Microsoft will only exist during a route leak/hijack). Among the multiple direct paths to Microsoft that a router may learn when its AS connects to Microsoft in different locations, the decision will usually fall to lowest IGP cost, choosing the nearest egress into Microsoft. Microsoft collocates anycast sites with all its peering locations, and so the nearest egress will often (and, in the case of the largest ring, always) be collocated with the nearest anycast site, aligning early exit routing with global optimization in a way that is impossible in the general case or with longer AS paths [70]. At smaller ring sizes, Microsoft can use traffic engineering (for example, not announcing to particular ASes at particular peering points) when it observes an AS making poor routing decisions.

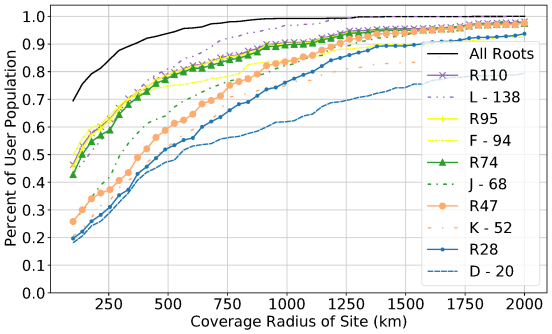
7.2 Larger Deployments are Less Efficient but Have Lower Latency

CDN latency in Figure 4a and inflation in Figure 5 reveal a relationship that some may find non-intuitive – as deployment size

[6]The plot is a box-and-whisker, with the 5 horizontal lines from bottom to top for each ⟨`deployment, AS path length`⟩ representing minimum, first quartile, median, third quartile, and maximum values.



(a)



(b)

Figure 7: Larger deployments lead to lower latency (Fig. 7a-left) since they offer more low-latency options to users (Fig. 7b). However, fewer users visit their closest site (Fig. 7a-right) leading to more inflation.

increases, inflation increases (less efficiency) but median latency decreases. We observe a similar effect in Figure 2 – larger root deployments tend to have more inflation but have lower latency. Intuitively, larger deployments are less efficient since BGP will make "wrong" decisions about which routes to export more often, and have lower latency since there are more low-latency options available to users. These results suggest efficiency may not be a useful metric for assessing performance.

We make these relationships explicit in Figure 7a which shows median latency and efficiency for each root letter and Microsoft ring. We define efficiency as the percentage of users with zero geographic inflation (*i.e.,* y-axis intercepts in Figure 2a and Figure 5a) since it is a rough measure of how optimal routing is (routing may not actually be optimal even if there is zero geographic inflation if users take a circuitous route to their closest site). Latency to root letters in Figure 7a is the median latency across all RIPE Atlas probes over an hour in 2018 (time of DITL) (*i.e.,* median per probe, then a median across probes), and latencies to rings are medians in Figure 4a.

The trend that efficiency decreases with deployment size is less clear in the root DNS than in Microsoft's CDN, likely since the root letters are run by different organizations and so have different deployment strategies which also impact latency and inflation. A counterexample to the trend is F root which had the lowest median latency (15 ms) in 2018 and good efficiency (39%). F root likely bucks the trend since F root partners with Cloudflare (a global CDN) and so benefits from a deployment tuned to lowering user latency. It is interesting that R95 and F root have similar number of

Table 1: Survey results from root DNS operators. Most root letters indicate DDoS resilience and (surprisingly) latency have been major factors for growth, and that future growth will likely slow.

Past		Future	
Reason for Growth	**Number of Orgs**	**Future Growth Trend**	**Number of Orgs**
Latency	8	Acceleration of Growth	1
DDoS Resilience	9	Decceleration of Growth	4
ISP Resilience	5	Maintain Growth Rate	4
Other	3	Cannot Share	1

sites and (low) median latency (approximately 15 ms), but that F root has considerably lower efficiency; hence, low efficiency is not necessarily bad. Conversely, high efficiency does not result in low latency; for example, 49% of users reach their closest B root site, but users still experience a high median latency to B root of 160 ms. Prior work looked at similar metrics to those in Figure 7a (right) for root letters using data from RIPE Atlas and arrived at very different conclusions [51], possibly since RIPE Atlas has limited coverage.

Part of what contributes to low latency is that organizations place sites close to users. Figure 7b shows what percent of Microsoft users are "covered" by a site in each ring and in a root letter of similar size, where "covered" means the closest site is within X km of users (x-axis). Hence, coverage implies there is a reasonably low latency option for users. Figure 7b is quite surprising – first, the root DNS as a whole (All Roots) has impressive coverage – 91% of Microsoft users are within 500 km of a root site (not even counting local sites!). Moreover, individual root letters can have even better coverage of Microsoft users than rings (L root has 94% of users within 1,000 km whereas R110 has 90%), which is interesting since L root, unlike R110, was not deployed specifically for Microsoft users. Figure 7b also demonstrates that approximating root DNS users with Microsoft users (Fig. 2) was fair, since root letters have decent coverage of Microsoft users. An exception is D root which did not have global sites in India at the time, where Microsoft has both anycast sites and a large user population to serve.

7.3 Differing Incentives Lead to Different Investments and Outcomes

We now discuss how incentives have shaped deployments and how our findings may extend to other anycast deployments.

7.3.1 Drivers for Growth. We reached out to operators of both root DNS and Microsoft asking what fueled their recent growth and whether they think it will continue. Of the twelve organizations running a root DNS letter, 11 responded, and we summarize the main reasons root DNS letters expand in Table 1. Principally, roots grew to reduce latency and improve DDoS resilience.

Over the past 5 years the number of root DNS sites has more than doubled from 516 to 1367, steadily increasing. Surprisingly, Table 1 demonstrates latency *was* a primary reason for expansion for nearly all root letters. Our results suggest this reasoning does not stem from caring about user experience (§4.3) but perhaps from establishing a competitive benchmark with other root letters.

Root operators also indicated growth was driven to improve resilience in two dimensions: DDoS and "ISP" resilience. DDoS resilience refers to increasing overall capacity so root letters can provide service in the face of DDoS attacks. ISP resilience refers to offering root sites in certain locations and networks so that service can still be offered even if connectivity to the rest of the Internet is

severed. According to both operator responses and publicly available sources, growth additionally stems from open hosting policies [40, 62, 74] (almost any AS can volunteer to host a new site) and from teaming up with large CDNs like Cloudflare. Root operator responses about future plans for growth suggest that the increase of root DNS sites will slow in the coming years.

With such decentralized deployment (in part by design to promote resilience), coordinated optimization of root DNS latency is difficult, even if latency optimization were a goal. By contrast, Microsoft's CDN is latency-sensitive and is centrally run. Operators optimize and monitor latency, thereby minimizing inflation (§6) with direct paths to many users (§7.1). Unlike some root letters, Microsoft does not (externally) compare latency with other CDNs, considering latency proprietary. Construction of new front-ends often follows business needs to support new markets. These commercial motivations contrast with the above root DNS reasons for expansion, yet the number of front-ends for Microsoft's CDN has more than doubled in the past five years.

7.3.2 Other Anycast Systems. A key takeaway from our results is that one cannot generalize our results to other systems using anycast. Anycast must be assessed in the context of the system in which it resides. Prior work took the results of one system (root DNS) and assumed it applied generally to a technique (anycast) which resulted in misleading conclusions [51]. It would be difficult to even extend our results to systems with similar deployments, since the degree to which performance improvements are due to the deployment and the degree to which they are due to tuning of route configurations is unknown [9].

Other systems using anycast include Akamai DNS authoritative resolvers [1], Google Cloud VMs [32], and Google Public DNS [31]. All of these services have different performance requirements for users; *i.e.,* they all want inflation to be "low" but how "low" it needs to be depends on the application. For example, Google Cloud VMs can host game engines which have much stricter latency requirements than fetching HTTP objects. We hope future work will take these considerations into account when assessing anycast.

8 RELATED WORK

Root DNS Anycast. Many prior studies look at latency and inflation performance in the root DNS [13, 51, 52, 67, 69]. Our work builds on these studies, conducting analysis for nearly every root letter and calculating inflation for millions of recursives in 35,000 ASes. These larger scale measurements offer broad coverage, enable comparisons among root letter deployments, and allow us to assess inflation in the root DNS *system* as a whole. We also calculate latency inflation differently than in prior work, which we believe offers a useful, orthogonal picture of inflation, and calculate inflation using the same methodology for both Microsoft's and root DNS, which allows us to compare inflation directly between Microsoft's CDN and root DNS (not possible with prior studies). Finally, we place latency and inflation in the context of user experience, while prior work on the root DNS does not. Other prior work looks at anycast's ability defend against DDoS attacks [58, 67]; we do not consider anycast's performance in this context. Other prior work discussed how ad-hoc anycast deployments can lead to poor performance and load balancing and is an early study of inflation in the root DNS [13]. Our work supports these conclusions and uses them in a larger conversation about anycast in the context of applications. We also confirm observations in prior work that anycast site affinity is high [12], at least over the duration of DITL.

CDN Anycast. Some CDNs use IP anycast [16, 21, 30, 65, 75]. Some prior work looked at inflation in CDNs [16], finding it to be similarly low. Our work presents a much larger study of latency and inflation (more than twice as many front-ends, orders of magnitude more users and measurements), updating the numerical results and lending confidence to the result that inflation is low; places performance metrics in the context of user experience; compares performance to other systems that use anycast; and provides some evidence of how CDNs can keep inflation low. Other prior work looked at how prefix announcement configurations can impact the performance of an anycast CDN [54]. More recent work has investigated how to diagnose and improve anycast performance through measurements in production systems [17, 43, 76]. Concurrent work examined addressing challenges faced by CDNs, proposing a scheme to decouple addressing from services that is compatible with anycast [27]. Our work characterizes, rather than changes, anycast CDN performance.

Recursive Resolvers, The Benefits of Caching, and Web Performance. Prior work has looked at statistics and latency implications of local resolvers [18, 44]. We calculate similar statistics using recent data. Some previous work looked at certain pathological behaviors of popular recursives and the implications these behaviors have on root DNS load times [34, 49, 73, 81]. We present additional pathological behavior of a popular recursive in Appendix E. Many studies characterize web performance and consider DNS's role in a page load [8, 11, 72], although none consider how root DNS specifically contributes to page load time and how this relates to user experience. Recent work considers placing DNS in the context of other applications but does not look at root DNS latency in particular [6].

9 CONCLUSION

While anycast performance is interesting in its own right, prior studies have drawn conclusions primarily from anycast for root DNS [51]. We have shown that anycast operates differently in CDNs, with less inflation. Differences stem from the impact the anycast service's latency and inflation has on user-perceived latency. Our results show the importance of considering multiple subjects in measurement studies and suggest why anycast continues to see wide, growing deployment.

Acknowledgements. This paper has been partially funded by NSF CNS-1835253 and NSF CNS-1836872. John Heidemann's work was supported in part by NSF CNS-1925737 and OAC-1739034. We would like to thank our shepherd Xiaowei Yang and the anonymous reviewers for their insightful comments, root DNS operators for their feedback on our analysis, and Dave Levin and Marcel Flores for their detailed feedback on an early draft of the paper.

REFERENCES

[1] Akamai. 2020. Designing DNS for Availability and Resilience Against DDoS Attacks. akamai.com/us/en/multimedia/documents/white-paper/akamai-designing-dns-for-availability-and-resilience-against-ddos-attacks.pdf

[2] Akamai. 2021. Akamai Compliance Programs. akamai.com/us/en/about/compliance/

[3] Mehmet Akcin. 2015. Comparing Root Server Performance Around the World. thousandeyes.com/blog/comparing-dns-root-server-performance

[4] Adiel Akplogan, Roy Arends, David Conrad, Alain Durand, Paul Hoffman, David Huberman, Matt Larson, Sion Lloyd, Terry Manderson, David Soltero, Samaneh Tajalizadehkhoob, and Mauricio Vergara Ereche. 2020. Analysis of the Effects of COVID-19-Related Lockdowns on IMRS Traffic. (April 2020). icann.org/en/system/files/files/octo-008-en.pdf

[5] Mark Allman. 2019. On Eliminating Root Nameservers from the DNS. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks (HOTNETS)* (Princeton, NJ, USA). ACM.

[6] Mark Allman. 2020. Putting DNS in Context. In *Proceedings of the 2020 Internet Measurement Conference (IMC)* (Online). ACM.

[7] Amazon. 2020. Amazon Route 53 FAQs. aws.amazon.com/route53/faqs/

[8] Internet Archive. 2020. The HTTP Archive Project. httparchive.org/

[9] Todd Arnold, Matt Calder, Italo Cunha, Arpit Gupta, Harsha V. Madhyastha, Michael Schapira, and Ethan Katz-Bassett. 2019. Beating BGP is Harder than we Thought. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks (HOTNETS)* (Princeton, NJ, USA). ACM.

[10] Todd Arnold, Ege Gürmeriçliler, Georgia Essig, Arpit Gupta, Matt Calder, Vasileios Giotsas, and Ethan Katz-Bassett. 2020. (How Much) Does a Private WAN Improve Cloud Performance? In *INFOCOM* (Online). IEEE.

[11] Alemnew Sheferaw Asrese, Pasi Sarolahti, Magnus Boye, and Jorg Ott. 2016. WePR: A Tool for Automated Web Performance Measurement. In *2016 IEEE Globecom Workshops* (Washington D.C., USA). IEEE.

[12] Hitesh Ballani and Paul Francis. 2005. Towards a Global IP Anycast Service. In *Proceedings of the 2005 ACM SIGCOMM Conference* (Philadelphia, PA, USA). ACM.

[13] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. 2006. A Measurement-Based Deployment Proposal for IP Anycast. In *Proceedings of the 2006 Internet Measurement Conference (IMC)* (Rio de Janeiro, Brazil). ACM.

[14] Ray Bellis. 2015. Researching F-root Anycast Placement Using RIPE Atlas. labs.ripe.net/author/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas/

[15] CAIDA. 2020. Inferred AS to Organization Mapping Dataset. caida.org/data/as-organizations/

[16] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *Proceedings of the 2015 Internet Measurement Conference (IMC)* (Tokyo, Japan). ACM.

[17] Matt Calder, Ryan Gao, Manuel Schröder, Ryan Stewart, Jitendra Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. 2018. Odin: Microsoft's Scalable Fault-Tolerant CDN Measurement System. In 15th *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (Renton, WA, USA). USENIX.

[18] Thomas Callahan, Mark Allman, and Michael Rabinovich. 2013. On Modern DNS Behavior and Properties. *ACM SIGCOMM Computer Communication Review* (July 2013).

[19] Neal Cardwell, Stefan Savage, and Tom Anderson. 2000. Modelling TCP Latency. In *INFOCOM* (Tel-Aviv, Israel). IEEE.

[20] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. 2015. Are We One Hop Away from a Better Internet? In *Proceedings of the 2015 Internet Measurement Conference (IMC)* (Tokyo, Japan). ACM.

[21] Danilo Cicalese, Jordan Augé, Diana Joumblatt, Timur Friedman, and Dario Rossi. 2015. Characterizing IPv4 Anycast Adoption and Deployment. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)* (Heidelberg, Germany). ACM.

[22] Cloudflare. 2020. What is DNS? cloudflare.com/learning/dns/what-is-dns/

[23] Lorenzo Colitti, Erik Romijn, Henk Uijterwaal, and Andrei Robachevsky. 2006. Evaluating the Effects of Anycast on DNS Root Name Servers. *RIPE Document RIPE-393* (Oct. 2006).

[24] Gerald Combs. 2020. Tshark. wireshark.org/docs/man-pages/tshark.html

[25] Team Cymru. 2020. IP to ASN Mapping Service. team-cymru.com/community-services/ip-asn-mapping/

[26] DNS-OARC. 2018. A Day in the Life of the Internet. dns-oarc.net/oarc/data/ditl/2018

[27] Marwan Fayed, Lorenz Bauer, Vasileios Giotsas, Sami Kerola, Marek Majkowski, Pavel Odinstov, Jakub Sitnicki, Taejoong Chung, Dave Levin, Alan Mislove, Christopher A. Wood, and Nick Sullivan. 2021. The Ties that un-Bind: Decoupling IP from Web Services and Sockets for Robust Addressing Agility at CDN-Scale. In *Proceedings of the 2021 ACM SIGCOMM Conference* (Online). ACM.

[28] Hongyu Gao, Vinod Yegneswaran, Jian Jiang, Yan Chen, Phillip Porras, Shalini Ghosh, and Haixin Duan. 2014. Reexamining DNS from a Global Recursive Resolver Perspective. *IEEE/ACM Transactions on Networking* (Oct. 2014).

[29] Manaf Gharaibeh, Han Zhang, Christos Papadopoulos, and John Heidemann. 2016. Assessing Co-locality of IP Blocks. In *Proceedings of 19th IEEE Global Internet Symposium* (San Francisco, CA, USA). IEEE.

[30] Danilo Giordano, Danilo Cicalese, Alessandro Finamore, Marco Mellia, Maurizio Munafò, Diana Zeaiter Joumblatt, and Dario Rossi. 2016. A First Characterization of Anycast Traffic from Passive Traces. In *Network Traffic Measurement and Analysis Conference (TMA)* (Louvain la Neuve, Belgium). IFIP/ACM.

[31] Google. 2020. Google Public DNS. developers.google.com/speed/public-dns

[32] Google. 2021. Cloud Load Balancing. cloud.google.com/load-balancing

[33] GTmetrix. 2019. The Top 1,000 Sites on the Internet. gtmetrix.com/top1000.html

[34] Wes Hardaker. 2020. What's in a Name? blog.apnic.net/2020/04/13/whats-in-a-name/

[35] John Heidemann, Katia Obraczka, and Joe Touch. 1997. Modelling the Performance of HTTP Over Several Transport Protocols. *ACM/IEEE Transactions on Networking* (Oct. 1997).

[36] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving High Utilization with Software-Driven WAN. In *Proceedings of the 2013 ACM SIGCOMM Conference* (Hong Kong). ACM.

[37] Geoff Huston. 2014. How Big is that Network? labs.apnic.net/?p=526

[38] IANA. 2020. IANA IPv4 Special-Purpose Address Registry. iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

[39] IANA. 2020. Root Servers. root-servers.org

[40] ICANN. 2020. Packet Clearing House. icannwiki.org/Packet_Clearing_House

[41] MaxMind Inc. 2020. IP Geolocation. maxmind.com/en/geoip2-databases

[42] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, and Min Zhu. 2013. B4: Experience with a Globally-Deployed Software Defined WAN. In *Proceedings of the 2013 ACM SIGCOMM Conference* (Hong Kong). ACM.

[43] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N Padmanabhan, Manuel Schroder, Matt Calder, and Arvind Krishnamurthy. 2019. Zooming in on Wide-Area Latencies to a Global Cloud Provider. In *Proceedings of the 2019 ACM SIGCOMM Conference*. ACM.

[44] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. 2002. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Transactions on networking* (Feb. 2002).

[45] Dina Katabi and John Wroclawski. 2000. A Framework for Global IP-Anycast (GIA). In *Proceedings of the 2000 ACM SIGCOMM Conference* (Stockholm, Sweden). ACM.

[46] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. 2006. Towards IP Geolocation Using Delay and Topology Measurements. In *Proceedings of the 2006 Internet Measurement Conference (IMC)* (Rio de Janeiro, Brazil). ACM.

[47] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. 2009. Moving Beyond End-to-End Path Information to Optimize CDN Performance. In *Proceedings of the 2009 Internet Measurement Conference (IMC)* (Chicago, IL, USA). ACM.

[48] W. Kumari and P. Hoffman. 2020. *Running a Root Server Local to a Resolver*. Technical Report 8806. Internet Request For Comments. www.rfc-editor.org/rfc/rfc8806.txt

[49] Matthew Lentz, Dave Levin, Jason Castonguay, Neil Spring, and Bobby Bhattacharjee. 2013. D-mystifying the D-root Address Change. In *Proceedings of the 2013 Internet Measurement Conference (IMC)* (Barcelona, Spain). ACM.

[50] Zhihao Li. 2019. *Diagnosing and Improving the Performance of Internet Anycast*. Ph.D. Dissertation. University of Maryland, College Park.

[51] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet Anycast: Performance, Problems, & Potential. In *Proceedings of the 2018 ACM SIGCOMM Conference* (Budapest, Hungary). ACM.

[52] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, and Jianping Wu. 2013. Measuring Query Latency of Top Level DNS Servers. In *International Conference on Passive and Active Network Measurement (PAM)* (Hong Kong). Springer.

[53] Zhuoqing Morley Mao, Charles Cranor, Fred Douglis, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. 2002. A Precise and Efficient Evaluation of the Proximity Between Web Clients and their Local DNS Servers. In *USENIX Annual Technical Conference* (Monterey, CA, USA). USENIX.

[54] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. 2019. Taming Anycast in the Wild Internet. In *Proceedings of the 2019 Internet Measurement Conference (IMC)* (Amsterdam, Netherlands). ACM.

[55] Christopher Metz. 2002. IP Anycast Point-To-(Any) Point Communication. *IEEE Internet Computing* (Aug. 2002).

[56] P. Mockapetris. 1987. Domain Names - Implementation and Specification. ietf.org/rfc/rfc1035.txt

[57] Giovane C. M. Moura, John Heidemann, Wes Hardaker, Jeroen Bulten, Joao Ceron, and Cristian Hesselman. 2020. Old But Gold: Prospecting TCP to Engineer DNS Anycast (extended). *ISI-TR-740, USC/Information Sciences Institute, Tech. Report* (2020).

[58] Giovane C. M. Moura, Ricardo de Oliveira Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the 2016 Internet Measurement Conference (IMC)* (Santa Monica, CA, USA). ACM.

[59] Mozilla. 2020. Window: Load Event. developer.mozilla.org/en-US/docs/Web/API/Window/load_event

[60] Moritz Müller, Giovane C. M. Moura, Ricardo de Oliveira Schmidt, and John Heidemann. 2017. Recursives in the Wild: Engineering Authoritative DNS Servers. In *Proceedings of the 2017 Internet Measurement Conference (IMC)* (London, United Kingdom). ACM.

[61] RIPE NCC. 2006. Evaluating The Effects Of Anycast On DNS Root Nameservers. ripe.net/publications/docs/ripe-393#efficiency

[62] RIPE NCC. 2018. Hosting a K-root Node. ripe.net/analyse/dns/k-root/hosting-a-k-root-node

[63] OpenDNS. 2020. Data Center Locations. opendns.com/data-center-locations/

[64] Craig Partridge, Trevor Mendez, and Walter Milliken. 1993. Host Anycasting Service. tools.ietf.org/html/rfc1546

[65] Matthew Prince. 2013. Load Balancing without Load Balancers. blog.cloudflare.com/cloudflares-architecture-eliminating-single-p/

[66] Jan Rüth, Christian Bormann, and Oliver Hohlfeld. 2017. Large-Scale Scanning of TCP's Initial Window. In *Proceedings of the 2017 Internet Measurement Conference (IMC)* (London, United Kingdom).

[67] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. 2006. On the Use of Anycast in DNS. In *Proceedings of the ACM SIGMETRICS Conference* (Banff, Canada). ACM.

[68] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *Proceedings of the 2017 ACM SIGCOMM Conference* (Los Angeles, CA, USA). ACM.

[69] Ricardo de Oliveira Schmidt, John Heidemann, and Jan Harm Kuipers. 2017. Anycast Latency: How Many Sites Are Enough? In *International Conference on Passive and Active Network Measurement (PAM)* (Sydney, Australia). Springer.

[70] Neil Spring, Ratul Mahajan, and Thomas Anderson. 2003. The Causes of Path Inflation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (Karlsruhe, Germany). ACM.

[71] RIPE NCC Staff. 2015. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal* (2015).

[72] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. 2013. Web Performance Bottlenecks in Broadband Access Networks. In *Proceedings of the ACM SIGMETRICS Conference* (Pittsburgh, PA, USA). ACM.

[73] Matthew Thomas. 2020. Chromium's Impact on Root DNS Traffic. blog.apnic.net/2020/08/21/chromiums-impact-on-root-dns-traffic

[74] Verisign. 2021. FAQ on RIRS Node Hosting. verisign.com/en_US/domain-names/internet-resolution/node-hosting/index.xhtml

[75] Verizon. 2020. verizondigitalmedia.com/media-platform/delivery/network/

[76] Lan Wei, Marcel Flores, Harkeerat Bedi, and John Heidemann. 2020. Bidirectional Anycast/Unicast Probing (BAUP): Optimizing CDN Anycast. In *Network Operations and Management Symposium* (Online). IEEE/IFIP.

[77] Lan Wei and John Heidemann. 2017. Does Anycast Hang Up on You? In *Network Traffic Measurement and Analysis Conference (TMA)* (Dublin, Ireland). IFIP/ACM.

[78] Florian Wohlfart, Nikolaos Chatzis, Caglar Dabanoglu, Georg Carle, and Walter Willinger. 2018. Leveraging Interconnections for Performance: The Serving Infrastructure of a Large CDN. In *Proceedings of the 2018 ACM SIGCOMM Conference* (Budapest, Hungary). ACM.

[79] Young Xu. 2017. 2017 Update: Comparing Root Server Performance Globally. thousandeyes.com/blog/2017-update-comparing-root-server-performance-globally/

[80] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proceedings of the 2017 ACM SIGCOMM Conference* (Los Angeles, CA, USA). ACM.

[81] Yingdi Yu, Duane Wessels, Matt Larson, and Lixia Zhang. 2012. Authority Server Selection in DNS Caching Resolvers. *ACM SIGCOMM Computer Communication Review* (April 2012).

[82] Yaping Zhu, Benjamin Helsley, Jennifer Rexford, Aspi Siganporia, and Sridhar Srinivasan. 2012. LatLong: Diagnosing Wide-Area Latency Changes for CDNs. *IEEE Transactions on Network and Service Management* (2012).

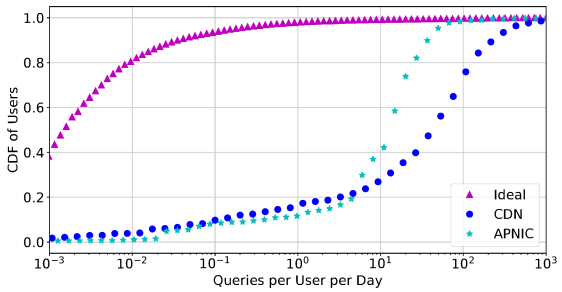


Figure 8: Daily queries by users to the root DNS, calculated by amortizing root DNS requests over user populations, when including or excluding queries for invalid TLDs. Counting invalid queries drastically increases median daily query counts to 22 (CDN), a 20-fold increase, or to 6 (APNIC), a 6-fold increase, depending on which user data we use.

Appendices are supporting material that has not been peer-reviewed.

A SUMMARY OF DATA

We use a myriad of datasets in the work, which is a result of our presenting answers to the questions we pose in several different ways (each with strengths and weaknesses). This approach allows us to overcome the limitations of individual datasets, by combining multiple views with different tradeoffs. To aid in comprehensibility, we summarize each of our datasets in Table 2 and Table 3.

As an example of how we use multiple views with different tradeoffs, consider the differences between the DITL packet traces (containing 51.9 billion queries across 50,000 ASes) and our local DNS / activity measurements (10 thousand measurements, 2 users). DITL allows us to see, globally, how recursive resolvers interact with the root DNS, allowing us to make definitive statements about global inflation and query volumes. However, DITL does not tell us how individual users interact with the root DNS, and translating DITL queries to user experience requires heuristic arguments about caching (§4.3). Our local DNS and activity measurements, although limited, give us precise reference points for how root DNS factors into everyday Internet browsing experience, which we find valuable.

B QUANTIFYING THE IMPACT OF METHODOLOGICAL DECISIONS

When analyzing latency and inflation, we often make assumptions or choose to conduct analysis a certain way. In what follows, we justify our various assumptions and pre-processing steps, and analyze the effects of these assumptions on our results.

B.1 Effect of Removing Invalid TLD Queries

In Section 4 we estimate the number of queries users experience due to the root DNS by amortizing queries over user populations. Out of 51.9 billion daily requests to all roots, we observe 31 billion daily requests for bogus domain names and 2 billion daily requests for PTR records. We choose to not count these towards user query counts, because we believe many of these queries do not lie on the critical path of user applications and so do not cause user-facing latency. This decision has a significant effect on conclusions we can draw, decreasing daily query counts to root DNS resolution by 20×.

We base this decision on prior work which investigated the nature of queries with invalid TLDs landing at the roots. ICANN has found that 28% of queries for non-existent domains at L root result from captive-portal detection algorithms in Chromium-based browsers [4]. Researchers at USC have found that more than 90% of single-string (not separated by dots) queries at the root match the Chromium captive-portal pattern [34]. We remove captive-portal detection queries from consideration since they occur on browser startup and network reconnect, not during regular browsing, and they can occur in parallel with browsing.

Some might argue that queries for invalid TLDs *are* associated with user latency because typos for URLs (when typing into a browser search bar, for example) cause users to generate a query to the root servers. However, typos only generate a query to the root server if the TLD is misspelled (as opposed to the hostname). Hence typos, in general, cause users latency, but only specific typos will cause users *root* latency. Moreover, prior work has found that approximately 60% of queries for invalid TLDs reaching root servers are for domains such as `local`, `no_dot`, `belkin`, and `corp` [28]. It is unlikely these queries are caused by typos, since they are actual (as opposed to misspelled) words and resemble domains often seen in software or in corporate networks. Chromium queries and queries for a certain set of invalid TLDs therefore account for around 86% of all queries for invalid TLDs at the roots, suggesting the vast majority of queries we exclude are not directly associated with user latency.

Nevertheless, it is still valuable to assess how including these queries for invalid TLDs changes the conclusions we can make about root DNS latency experienced by users. Figure 8 shows daily user latencies due to root DNS resolution when we include requests for invalid TLDs and PTR records in daily query volumes. Using CDN user counts, users experience a median of 22 queries to the root DNS each day – about 20× more than when we exclude requests for invalid queries (§4). This drastic 20-fold increase is surprising given we only (roughly) double the amount of queries by including invalid queries. The difference is best explained by the fact that a majority of invalid queries are generated by /24s with a large number of users. Since the y-axis of Figure 8 is the number of users (not /24s), counting invalid queries shifts the graph far to the right. Hence, counting invalid queries drastically affects the conclusions we can draw. There is a less severe 6-fold increase in the number of queries per user per data calculated using APNIC data. Overall, including invalid TLD queries drastically changes our quantitative conclusions about user interaction with the root DNS but may not change our qualitative conclusions, since 20 queries a day to the roots is still small.

B.2 Representativeness of Daily Root Latency Analysis

In Section 4 we estimate the number of queries users experience due to the root DNS by amortizing queries over user populations. To obtain estimates of user populations, we obtain counts of Microsoft users who use recursives (§2.1). Naturally recursives used by Microsoft users and recursives seen in DITL do not overlap perfectly.

Table 2: Summary of Datasets

Dataset	# of Measurements	Duration	Year	# of ASes	Technology/Format
Sampled CDN Server-Side Logs (§6)	11.0 $\times 10^9$	1 week	2019	59 000	Windows TCP/IP, HTTPService (TCP RTT)
Sampled CDN Client-Side Measurements (§5.2)	50.0 $\times 10^7$	1 week	2019	10 600	Odin [17] (HTTP GET)
CDN User Counts (§4.3)	—	1 month	2019	39 000	Custom URL DNS Requests
APNIC User Counts (§4.3)	—	updated daily	2019	23 000	Google Ad Delivery Network
DITL Packet Traces (§2.1)	51.9 $\times 10^9$	2 days	2018	50 300	Packet Traces
DITL ∩ CDN (§3, §4.3, §7)	18.6 $\times 10^9$	—	2018–2019	35 500	Root DNS query and user counts
RIPE Atlas (§5.2, §7.1)	10.0 $\times 10^3$	1 hour	Various	3 300	ping, traceroute
USC/ISI (§4.3)	10.0 $\times 10^7$	1 year	2018	1	Packet Traces
Local DNS / Activity Measurements (§4.3)	68.0 $\times 10^4$	1 month	2020	2	Packet Traces, Chrome Webtime Tracker

Table 3: Strengths and Weaknesses of Datasets

Dataset	Strengths	Weaknesses
Sampled CDN Server-Side Logs (§6)	Has client to front-end mappings, global coverage	Cannot hold user population fixed across rings
Sampled CDN Client-Side Measurements (§5.2)	Can hold user population fixed across rings, global coverage	Do not know which front-end the client reached, smaller scale
CDN User Counts (§4.3)	Precise estimates of user counts, global coverage	Under estimates user counts
APNIC User Counts (§4.3)	Global coverage, publicly accessible	Not validated, coarse granularity
DITL Packet Traces (§2.1)	Global coverage	Noisy, only above the recursive resolver
DITL ∩ CDN (§3, §4.3, §7)	Global coverage, attributes queries to users	Excludes v6
RIPE Atlas (§5.2, §7.1)	Historic data, reproducibility	Limited coverage
USC/ISI (§4.3)	Precise, below the recursive,	Limited coverage, no information about users
Local DNS / Activity Measurements (§4.3)	Precise, at the end user	Limited coverage, small scale

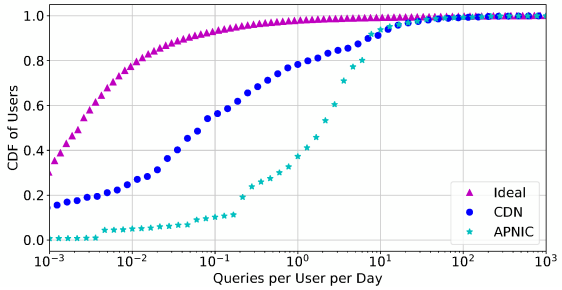


Figure 9: A CDF of the number of queries Microsoft users experience due to root DNS resolution, per day, without joining recursives by /24 in DITL with recursives seen by Microsoft (CDN). This unrepresentative analysis yields an estimate of daily user queries far, far lower than in Section 4.3.

Table 4: Statistics displaying the extent to which the recursives of users in Microsoft's CDN overlap recursives seen in the 2018 DITL captures without users and volumes by /24. Also shown in parentheses are corresponding statistics when joining by /24. Joining the datasets by /24 increases most measures of representation by tens of percents, with some measures increased by up to 64%.

dataset	Statistic	Percent Overlap (by /24)
DITL ∩ CDN	DITL Recursives	2.45% (29.3%) of DITL Recursives
	DITL Volume	8.4% (72.2%) of DITL Query Volume
	CDN Recursives	41.9% (78.8%) of CDN Recursives
	CDN Volume	47.05% (88.1%) of CDN Query Volume

To increase the representativeness of our analysis, we aggregate Microsoft user counts and DITL query volumes by resolver /24, and join the two datasets on /24 to create the DITL∩CDN dataset. The intuition behind this preprocessing step is that IP addresses in the same /24 are likely colocated, owned by the same organization, and act as recursives for similar sets of users. We now justify this decision and discuss the implications of this preprocessing step on the results presented in Section 4.3.

In Table 4 we summarize the extent to which the recursives seen by Microsoft are representative of the recursives seen in DITL, and vice-versa, without aggregating by /24. We also display corresponding statistics when aggregating by /24 for comparison in parentheses. Clearly joining by /24 makes a significant difference, increasing various measures of overlap by tens of percents and in certain cases by up to 64%.

As an analogy to Figure 3, in Figure 9 we show the number of queries each Microsoft user executes to the roots per day *without* aggregating query and user statistics by /24 (CDN). We also show APNIC as in Figure 3 for comparison, even though APNIC is not affected by /24 volume aggregation. Users of CDN only send 0.036 queries to the roots each day at the median – roughly one 30^{th} of the estimate obtained when aggregating statistics by /24. This small daily user latency makes sense, given that we only capture 8.4% of DITL volume without joining the datasets by /24 (Table 4).

Table 4 and Figure 9, demonstrate that the decision to aggregate statistics and join DITL captures with Microsoft user counts by /24 led to both much greater representativeness of the analysis and very different conclusions about user interactions with the root DNS. We would now like to justify this decision using measurements. If, as we assume, IP addresses in the same /24 are colocated, they are probably routed similarly. Prior work has shown that only a small fraction of anycast paths are unstable [77], and so we expect that, over the course of DITL, IP addresses in the same /24 reach the same anycast sites.

As a way of quantifying routing similarity in a /24, in Figure 10 we show the percent of queries from each /24 in DITL that do not reach the most "popular" anycast site for each /24 in each root deployment. We label root letters alongside the total number of

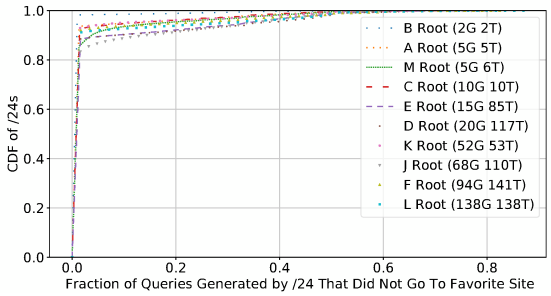


Figure 10: Fractions of queries generated by /24s that do not hit the most popular site for each /24 and for each root letter in question. The legend indicates the number of global sites (G) and total (global and local) sites (T). For all root letters, more than 80% of /24s have all queries visit the most popular site, suggesting queries from the same /24 are usually routed similarly.

sites (local and global) that they had during the 2018 DITL. For each root letter and for each /24 that queried that root letter in DITL, we look at how queries from the /24 are distributed among sites.

Let q_{ij}^k be the number of daily queries from IP i in /24 k toward anycast site j. We then calculate the fraction of queries that do not visit the most "popular" site as

$$f^k = 1 - \sum_i \frac{q_{ij_F}^k}{Q^k} \tag{3}$$

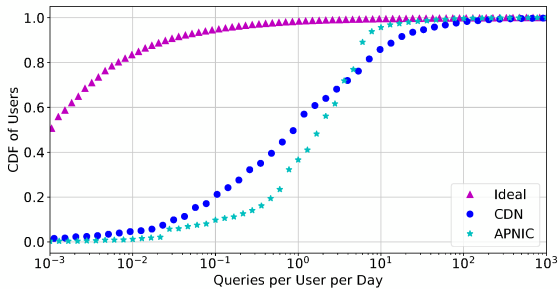
where j_F is the favorite site for /24 k (*i.e.,* the site the /24 queries the most), and Q^k is the total number of queries from /24 k. We plot these fractions for all /24s in DITL, and for each root deployment. (We do not include /24s that had only one IP from the /24 visit the root letter in question.)

For more than 80% of /24s, all queries visit only one site per root letter, suggesting that queries from the same /24 are routed similarly. This analysis is slightly biased by the size of the root deployment. For example, two IP addresses selected at random querying B root would hit the same site half the time, on average. However, even for L root, with 138 sites, more than 90% of /24s direct all queries to the most popular site. We believe Figure 10 provides evidence that recursives within the same /24 prefix are located near each other, and hence serve similar sets of users.

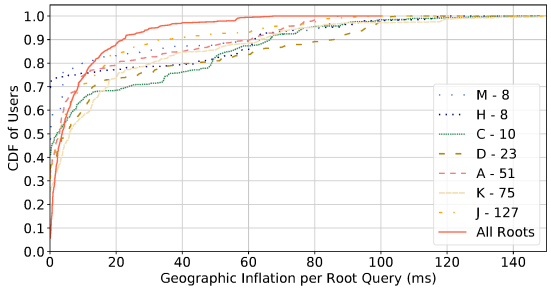
Even queries from a single IP address within a /24 may reach multiple sites for a single root over the course of the DITL captures. Such instability can make routing look less coherent across IP addresses in a /24, even if they are all routed the same way. Controlling for cases of changing paths for the same IP makes intra-/24 routing even more coherent. If we let the distribution of queries generated by an IP address to a root be a point mass, with all the queries concentrated at that IP addresses' favorite site, all queries from more than 90% of all /24s to all roots are routed to the same site (not shown).

B.3 Implications of Using the 2018 DITL

At the time of writing, the 2020 DITL was available to use in the study, but we chose to use the 2018 study since the 2018 study had better coverage of root letters. (Neither has perfect coverage – for both 2018 and 2020 DITLs, G root is not included and I root is completely anonymized.)



(a)



(b)

Figure 11: Queries per user per day to the root DNS and inflation of root letters calculated using the 2020 DITL. Our high level conclusions about how much inflation is in the root DNS and the number of queries users experience per day do not change depending on the year.

For the 2020 DITL specifically, B root was not available at the time of writing (but may be in the future), E root includes only one site (out of 132), F root does not include any Cloudflare sites (more than half the volume), and L root is completely anonymized (hence unusable). The 2018 DITL has none of these limitations, and so our results apply to more letters. Studying the root DNS system as a whole is a key strength of our analysis compared to prior work, so we feel coverage is more important than having the most up-to-date results for only a subset of root letters.

For completeness, and to demonstrate that our larger takeaways about root DNS latency and inflation do not change significantly from year to year, we calculate queries per day (as in Figure 3) and inflation (as in Figure 2) for the root letters for which we have data, and the results are shown in Figure 11.

Our high level conclusions about root DNS latency do not change when looking at the 2020 DITL – most users still experience about one DNS query per day, and the number of root queries sent by recursives is still far from the 'ideal' querying behavior of one record per TTL. Inflation results are also similar – individual root letters have less inflation (for example, D root improved). Average geographic inflation is almost exactly the same as in 2018, with approximately 10% of users experiencing more than 20 ms (2,000 km) of inflation.

C NUMBER OF RTTS IN A PAGE LOAD

To estimate the latency a user experiences when interacting with Microsoft's CDN (§5.2), we first estimate the number of RTTs required to load a typical web page hosted by Microsoft's CDN. The number of RTTs in a page load depends on a variety of factors, so we aim to find a reasonable *lower bound* on the number of RTTs users incur for typical pages. A lower bound on the number of RTTs to load pages is a conservative measure of the impact of CDN inflation, as latency inflation accumulates with each additional RTT, and larger pages (more RTTs) would be impacted more. We provide an estimate of this lower bound based on modeling and evaluation of a set of web pages hosted by Microsoft's CDN using Selenium (a headless web browser), finding that 10 RTTs is a reasonable estimate. We scale latency by the number of RTTs in Section 5.2 to demonstrate how improvements in latency help users (and, conversely, how inflation hurts users).

Users incur latency to Microsoft's CDN when they download web objects via HTTP. We calculate the number of RTTs required to download objects in each connection separately, and sum RTTs over connections while accounting for parallel connections. For a single TCP connection, the number of RTTs during a page load depends on the size of files being downloaded. This relationship is approximated by

$$N = \lceil log_2 \frac{D}{W} \rceil \tag{4}$$

where N is the number of RTTs, D is the total number of bytes sent by the TCP connection from the server to the user, and W is the initial congestion window size in bytes [19, 35]. Although W is set by the server, Microsoft and a majority of web pages [66] set this value to approximately 15 kB so we use this value. We do not consider QUIC or persistent connections across pages in detail here, but larger initial windows will result in fewer RTTs. We test mostly landing pages, for which persistent connections are uncommon. Moreover, such considerations likely would not change our qualitative conclusions about how users experience CDN latency.

We make the following assumptions to establish a *lower bound* on N: (1) we do not account for connections limited by the receive window or the application, as the RTT-based congestion window limitation we calculate is still a lower bound, (2) TCP is always in slow start mode, which implies the window size doubles each RTT and serves as a lower bound on the actual behavior of Microsoft's standard CUBIC implementation, and (3) all TCP and TLS handshakes after the first do not incur additional RTTs (*i.e.,* they are executed in parallel to other requests).

Modern browsers can open many TCP connections in parallel, to speed up page loads. Summing up RTTs across parallel connections could therefore drastically overestimate the number of RTTs experienced users. To determine the connections over which to accumulate RTTs, we first start by only considering the connection with the most data. We then iteratively add connections in size-order (largest to smallest) that do not overlap temporally with other connections for which we have accumulated RTTs. The 'data size' of a connection may represent one or more application-layer objects.

We load nine web pages owned by Microsoft, twenty times for each page. We choose popular pages hosted on Microsoft's CDN with dynamic content suggested to us by a CDN operator. We use Selenium and Chrome to open web pages and use Tshark [24] to capture TCP packets during the page load. When the browser's `loadEventEnd` event fires, the whole page has loaded, including all dependent resources such as stylesheets and images [59]. So, to calculate the total data size for each connection, we use the ACK value in the last packet sent to the server before `loadEventEnd` minus the SEQ value in the first packet received from the server. We then calculate the number of RTTs using Equation (4), and add a final two RTTs for TCP and TLS handshakes. We find only a few percent of CDN web pages are loaded within 10 RTTs, and 90% of all page loads are loaded within 20 RTTs, so 10 RTTs is a reasonable lower bound.

D LATENCY MEASUREMENTS AT A RECURSIVE RESOLVER

To obtain a local perspective of how users experience root DNS latency, we use packet traces from ISI. Here, we characterize DNS and root DNS latencies users experience at the resolver, along with a useful visualization of how inconsequential root DNS latency is for users at this resolver. This analysis complements our global view of how users interact with the root DNS in Section 4.3, as it demonstrates how often everyday users might send queries to the root relative to other DNS queries.

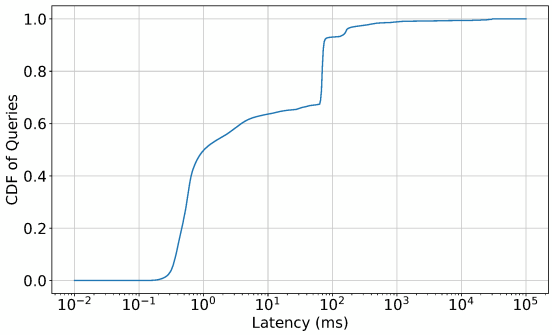


Figure 12: CDF of user DNS query latencies seen at a recursive resolve at ISI, over the course of one year. Latencies are measured from the timestamp when the recursive resolver receives a client query to the timestamp when the recursive sends a response to that client query. The sub-millisecond latency for more than half of queries suggests most queries to this recursive are served by the local cache.

Figure 12 shows the latencies of all queries seen at the recursive resolver over one year, where latencies are measured from the timestamp when the recursive resolver receives a client query to the timestamp when the recursive sends a response to that client query. Latencies are divided into (roughly) 3 regions: sub-millisecond latency, low latency (millisecond - tens of milliseconds), and high latency (hundreds of milliseconds). The first region corresponds to cached queries, so roughly half of queries are (probably) cached. The second region corresponds to DNS resolutions for which the resolving server was geographically close. Finally, the third region likely corresponds to queries that had to travel to distant servers, or required a few rounds of recursion to fully resolve the domain.

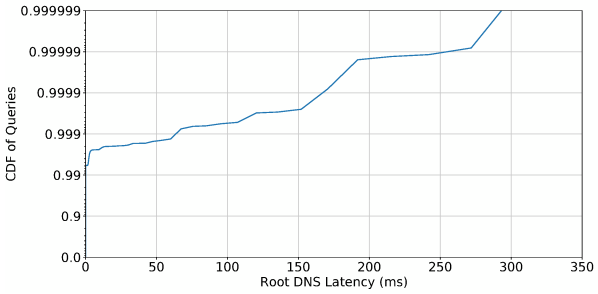


Figure 13: Root DNS latency for queries made by users of ISI recursive resolver during 2018. This plot demonstrates the benefits of caching and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than 0.1% incur latencies greater than 100 ms. User queries that did not generate a query to a root server were given a latency of 0.

The sub-millisecond latency for more than half of queries suggests most queries to this recursive are served by the local cache. These latencies are similar to those presented in previous work that also studied a recursive resolver serving a small user population [18]. Queries in the second and third regions include queries that did not query the root (since those records were cached) but did query other parts of the DNS hierarchy.

As discussed in Section 4, root DNS queries make up a small fraction of all queries shown in Figure 12. To visualize just how small this fraction is, Figure 13 shows a CDF of root DNS latency experienced for queries over 2018. Requests that do not generate a query to a root server are counted as having a root latency of 0. Figure 13 demonstrates the benefits of shared caches and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than 0.1% incur latencies greater than 100 ms.

E CASE STUDY: REDUNDANT ROOT DNS QUERIES

When we investigate the traffic from a recursive resolver to the root servers in Section 4, we see as many as 900 queries to the root server in a day for the COM NS record. Given the 2 day TTL of this record, this query frequency is unexpectedly large. This large frequency motivated us to analyze why these requests to roots occurred. We consider a request to the root to be redundant if a query for the same record occurred less than 1 TTL ago. Prior work has investigated redundant requests to root servers as well, and our analysis can be considered complementary since we discover different reasons for redundant requests [28].

To observe these redundant requests in a controlled environment, we deploy a BIND instance (the resolver in Section 2.1 runs BIND v9.11.17) locally and enable cache and recursion. We do not actually look up the cache of the local BIND instance to see which records are in it. Instead, we save the TTL of the record and the timestamp at which we receive the record to know if the record should be in BIND's cache. We use BIND version 9.11.18 and 9.16.1. Because 9.16.1 is one of the newest releases and 9.11.18 is a release from several years ago, we can assume that pathological behavior is common in all versions between these two releases. After deploying the instance, we simulate user behavior by opening the top-1000 web pages according to GTmetrix [33] using Selenium and headless Chrome. While loading web pages, we collect network packets on port 53 using Tshark [24].

For these page loads, we observe 69,215 DNS A & AAAA-type requests generated by the recursive resolver. 3,137 of these requests are sent to root servers, and 2,950 of these root DNS queries are redundant. Over 70% of redundant requests are AAAA-type. After investigating the cause of these redundant queries, we find over 90% of these redundant requests follow a similar pattern. This pattern is illustrated by the example in Table 5.

In Table 5, we show queries the recursive resolver makes when a user queries for the A record of bidder.criteo.com. In step 1, the recursive resolver receives a DNS query from a client. According to TTL heuristics, the COM A record is in the cache. In step 3, the TLD server responds with records of authoritative nameservers for "criteo.com". Then, the recursive chooses one of them to issue the following request to. However, for some reason (*e.g.*, packet loss), the recursive resolver does not get a response from the nameserver in step 4. Hence, the resolver uses another nameserver in step 5, which it learned in step 3. At the same time, as seen in step 6 to 11, the recursive sends (redundant) DNS requests to root servers, querying the AAAA-type records for these nameservers. These requests are redundant since the AAAA record for COM was received less than two days ago.

From the pattern demonstrated in Table 5, we hypothesize that redundant requests to the root servers will be generated for certain records when the following conditions are met.

(1) A query from the recursive resolver to an authoritative nameserver times-out.
(2) The record queried for by the resolver to the root DNS server was not included in the Additional Records section of the TLD's response.

The second condition is also why we were seeing more AAAA-type redundant requests, because usually there are more A-type records in the Additional Records section than AAAA-type records.

To see how much traffic is caused by our hypothesis in a real scenario, we analyze packet captures on a recursive resolver (BIND 9.11.17) serving users at ISI. To keep consistent with the other analysis we do on this dataset (§4), we use packet captures from 2018. 79.8% of requests to roots are redundant and in the pattern we described. The other 20.2% consists of necessary requests and requests for which we have no hypothesis as to how they were generated. We contacted developers at BIND, who said this may be a bug.

Software behavior as described here can lead to orders of magnitude more root DNS requests than would be necessary if recursives queried for the record once per TTL. As demonstrated in Figure 3, focusing on reducing the number of these queries could both improve user experience and reduce load on the root server.

F VISUALIZATION OF MICROSOFT CDN PERFORMANCE

In Section 2.2 we show the rings of a large anycast CDN and how users are distributed with respect to those rings. This visualization does not include any information about latency, so we provide one here. In Figure 14 we show front-ends in R110, and associated latency users experience to R110 in each region. Transparent circles

Table 5: Redundant root DNS requests. The last five requests to J root are redundant which may be caused by an unanswered request in step 4.

Step	Relative Timestamp (second)	From	To	Query name	Query type	Response
1	0.00000	client	resolver	bidder.criteo.com	A	
2	0.01589	resolver	192.42.93.30 (g.gtld)	bidder.criteo.com	A	
3	0.02366	192.42.93.30 (g.gtld)	resolver	bidder.criteo.com	A	ns23.criteo.com ns22.criteo.com ns25.criteo.com ns26.criteo.com ns27.criteo.com ns28.criteo.com.
4	0.02387	resolver	74.119.119.1 (ns25.criteo.com)	bidder.criteo.com	A	
5	0.82473	resolver	182.161.73.4 (ns28.criteo.com)	bidder.criteo.com	A	
6	0.82555	resolver	192.58.128.30 (j.root)	ns22.criteo.com	AAAA	
7	0.82563	resolver	192.58.128.30 (j.root)	ns23.criteo.com	AAAA	
8	0.82577	resolver	192.58.128.30 (j.root)	ns27.criteo.com	AAAA	
9	0.82584	resolver	192.58.128.30 (j.root)	ns25.criteo.com	AAAA	
10	0.82592	resolver	192.58.128.30 (j.root)	ns26.criteo.com	AAAA	
11	0.82620	resolver	192.58.128.30 (j.root)	ns28.criteo.com	AAAA	

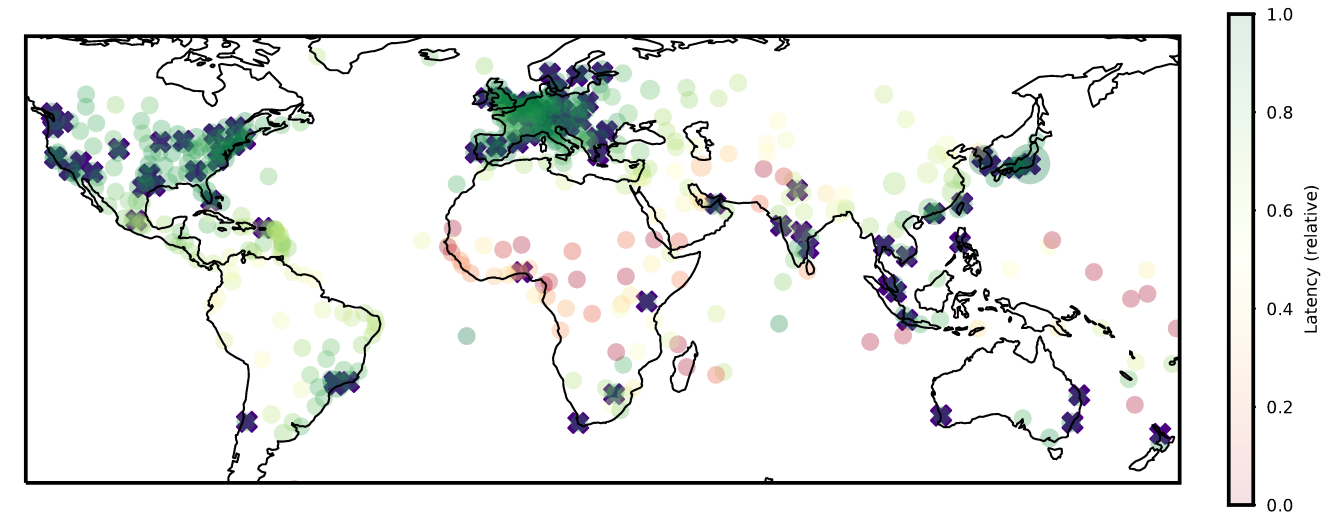


Figure 14: A visualization of front-ends in R110 (purple Xs), and user populations (transparent circles). User populations are colored according to the relative latency they experience and have size proportional to user population. Red corresponds to high latency, and green corresponds to low latency. Latency generally gets lower the closer users are to a front-end, and front-ends are concentrated around large user populations.

represent user populations and their radii are proportional to the user population. Population circles are colored according to average median latency users in the metro experience to R110 – red indicates higher latency while green indicates lower latency. Latency generally gets lower the closer users are to a front-end. The CDN has focused on deploying front-ends near large user populations, which has driven latencies quite low for nearly all users.