Anycast in Context: A Tale of Two Systems

Thomas Koch
Columbia University

Ethan Katz-Bassett
Columbia University

Matt Calder
Microsoft, Columbia University

John Heidemann
ISI

Ke Li
Columbia University

ABSTRACT

Anycast is widely used today to serve content of many types, from web pages to DNS. Research, often examining root DNS, has suggested that inflation is a problem for anycast, with users often routed to suboptimal sites. Nevertheless, anycast is widely used in practice by many large, latency-sensitive services. We reassess anycast performance, examining both DNS and CDNs. We first extend prior analysis on inflation in the root DNS, showing that inflation is very common in root DNS, affecting more than 95% of users. This finding may seem surprising, given the large, growing number of root DNS sites. We investigate how often users interact with the root DNS and find root DNS latency hardly matters to users because caching is so effective. These findings lead us to question: is inflation inherent to anycast, or does inflation occur when it does not matter to users? To answer this question, we consider a large, operational anycast CDN serving latency-sensitive content. Here, latency matters orders of magnitude more than for root DNS. Perhaps because of this need, we find that only 35% of CDN users see any inflation, and it is smaller than root DNS. We show that CDN anycast latency has little inflation due to extensive peering and engineering. Hence, the application matters – latency-sensitive applications like CDNs optimize user experience, but for root DNS, latency has minimal user impact. These results suggest prior claims of anycast inefficiency reflect experiments of a single application rather than its technical potential.

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) [4, 8, 10, 14, 59] and Content Delivery Network (CDN) [26, 31, 36, 59, 68] deployments today, in part because of its ability to improve latency to clients and decrease load on each anycast server [44, 53, 58].

However, studies have argued that anycast provides suboptimal performance for some users, compared to the lowest latency one could achieve given deployed sites [26, 49, 52, 61, 72]. Results have suggested that anycast routing is ineffective, failing to direct users to the lowest latency site [49, 50, 61]. Perhaps because of these studies, we have found the opinion of anycast among some experts in the community is poor. These results may seem surprising given continuing growth in production systems that use anycast – why continue to use anycast if it causes inflation?

To understand the actual impact of anycast efficiency, and its wide use in spite of inflation, we step back and evaluate anycast as a component of an actual application. User-affecting performance depends on the anycast deployment, anycast use within the system, and how users interact with the system. To see these effects we consider anycast's role within two real-world systems: the root DNS and a large 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 [14] packet traces which are available via DITL [1] and which are featured in existing anycast studies [33, 49, 52, 56, 63]. The 13 root letters operate independently with diverse deployment strategies, enabling the study of different anycast systems providing the same service. We analyze two days of unsampled packet captures from 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, representative coverage.

We also examine a major commercial CDN, which we call ACDN. ACDN 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 ACDN users in hundreds of countries, giving us a complete view of ACDN performance.

With these measurements, we present the largest study of anycast latency and inflation to date. We first validate and extend prior work [49] on anycast inflation. Whereas that work focused primarily on a single root instance, we analyze almost the whole root DNS. By joining root DNS captures with global-scale traces of user behavior, we find that root DNS queries are inflated for most users, and can be inflated by over a hundred milliseconds. In fact, we find inflation to the roots is worse than previously thought – on average, more than 95% of users experience some inflation to the roots.

These results hardly make sense, given the number of root DNS sites has more than doubled in the past five years

– why is performance so poor with so much investment in infrastructure? To answer this question, we question if *user* performance is in fact poor, by investigating how users interact with the root DNS. We find differences in latency and inflation among root letters are hardly perceived by users – using global-scale traces we show that 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 recursives.

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. To determine which is the case, we use measurements from ACDN and find that, were latency to the CDN to be inflated as in the roots, 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. With this context, we then investigate *actual* inflation in ACDN and find that inflation is kept comparatively small – about half that of the roots. To explain why inflation is so different in these systems, we contrast AS-level connectivity and inflation between the users, ACDN, and roots. We find that ACDN is able to control inflation through extensive peering and engineering investment (§7). Moreover, through discussions with operators of root DNS and CDNs, we find recent root DNS expansion is primarily for mitigating DDoS attacks while CDN expansion is driven by market forces (§7.2).

The comparison between performance in these two systems allows us to put results from prior work in perspective [26, 33, 49, 63]. Even though root inflation is large, users rarely experience it, making average inflation quite small. In contrast, users frequently interact with ACDN and inflation there is small. These inflation results make sense, given the economic incentives of the organizations running ACDN and the root DNS. We expect these results to hold for other CDNs besides ACDN, since other CDNs have similar economic incentives. 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; our study are anycast systems and not individuals. We will release our analysis source. Our CDN data is proprietary, but we are exploring if portions may be releasable.

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 [1], while CDN data is proprietary. We supplement these datasets with measurements from RIPE Atlas [65].

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 [32, 54]. To resolve a name to its result, a user sends DNS requests to recursive resolvers (recursives). 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 thirteen letters [14], each with a different anycast deployment with 6 to 254 anycast sites (as of January 2021), run by 12 commercial and non-profit organizations.

We use three datasets: for end-users, we use long-term packet captures from a research lab in a university in the United States, 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) [1].

Packet captures from a research lab in a university in the United States 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, 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.

To obtain a global view of root DNS use, we use the 2018 DITL captures, archived by DNS-OARC [1] (2018 is the most recent available). 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.

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 now describe this pre-processing of DITL and subsequent joining of root query volumes with ACDN 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 [19, 37, 67], and so involve machine startup and not browsing latency. Prior work suggests the remainder are generated by other malfunctioning, automated software [34]. Similarly, while PTR queries have some uses (traceroutes and confirming hostnames during authentication), they are not part of typical user web latency. In Appendix A.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 [12] (7% of all queries). Finally, we analyze only IPv4 data and exclude IPv6 traffic (12% of queries) because we lack v6 user data; we are working to get that for future research.

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 ACDN users of each recursive, gathered in 2019 (the oldest user data we have). This user data is from DNS data at ACDN, which counts unique IP addresses as "users". This definition undercounts multiple human users that use a single IP address with Network Address Translation. ACDN maps recursives to user IP addresses with an existing technique that instruments users to request DNS records for domains ACDN controls when users interact with ACDN [27, 51].

We join the DITL captures and ACDN user counts by the recursive resolver /24, aggregating DITL query volumes and ACDN 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 [4, 9]. Prior work has also found that up to 80% of /24's are collocated [35]. We provide additional justification for this preprocessing step in Appendix A.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 data set of query volumes and user counts by recursive DITL∩ACDN.

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 [40] (*i.e.,* instead of using proprietary ACDN data). These AS user population estimates were obtained by first gathering lists of IP addresses from Google's Ad delivery network, separated by country. This distribution of IP addresses was converted to a distribution of ASNs, and normalized by country Internet-user populations. The major assumptions made by APNIC when generating user counts are that Google Ad placement is proportional to an AS's user base within a country, and that both an AS and its users are fully contained in a single country. We use the TeamCymru IP to ASN mapping to map IP addresses seen in the DITL captures to their respective ASes [5], and accumulate queries by ASN. We were able to map 99.4% of IP addresses (seen in DITL) 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 ACDN user counts are more accurate, but APNIC data is more accessible to other researchers and so provides a useful comparison.

[1]We take care to ensure that user counts are not "double counted" across different resolver IP addresses in the same /24.

2.2 ACDN

We also study ACDN, a large anycast CDN that serves web content to over a billion users from more than 100 sites. Traffic destined for ACDN enters its network at a point of presence (PoP), and is routed to one of the anycast sites serving the content (front-ends). ACDN 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. Other CDNs have to work with similar regulatory restrictions [18]. Hence, traffic from a user prefix destined for ACDN may end up at different front ends (depending on the application), 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.

ACDN operates multiple anycast rings providing different size anycast deployments for study. In Figure 1 we show the front-ends and user concentrations of ACDN. 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). 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 deployment locations tend to be near large populations, providing at least one low latency option to most users. Appendix D illustrates latency differences by region.

User locations are aggregated by *region*, a geographic area used internally by ACDN 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⟩ are often routed to the same front-ends and so (generally) experience similar latency. There are 508 regions in total: 135

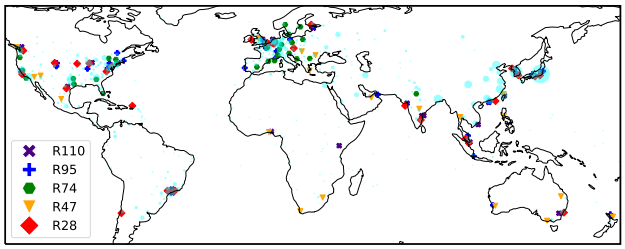


Figure 1: The hierarchical ring structure, and its global presence. User populations are shown as circles, with the radius of the circle proportional to the number of users in that region, demonstrating that ACDN has deployed front-ends in areas of user concentration.

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 ACDN, 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⟩ to each front-end that serves them.[2] ACDN determines the location and AS of users using internal databases.

Client-side measurements come from a measurement system operated by ACDN, similar to those described in the literature [27, 51]. Latency measurements are the time it takes for clients of ACDN to fetch a small image via HTTP.[3] The measurement system instructs clients using CDN services to issue latency 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). ACDN collects latencies of users populations, noting the location and AS of the user. Since these measurements come directly from end-users, however, we do not know which front-end the user hit. For both client-side measurements and server-side logs, we collect statistics of over a billion users across 15,000 ⟨region, AS⟩ locations.

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

[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.

3 ROUTES TO ROOT DNS ARE INFLATED

Earlier work has found query distance to the root DNS is often significantly inflated [25, 33, 49, 61, 63]. 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 inflation for recursives rather than users, we conduct the analysis for nearly all root letters, and place inflation in the context of *users*, rather than recursive resolvers. These contributions are significant for two reasons. First, considering more root letters allows us to evaluate inflation in additional, different deployments, and with most letters we can evaluate the root DNS *system*, rather than one root letter. Since a recursive makes queries to many root letters, favoring those nearby [57], *system* performance and inflation can 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 [25, 26, 49, 63]. We choose to take a holistic approach to studying inflation, considering inflation relative to the deployment, rather than trying to infer which inflation would exist in an equivalent unicast deployment for several reasons. First, coverage in systems such as RIPE Atlas (vantage points for anycast studies [49, 63]) is not necessarily representative [22]. Second, calculating unicast path 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. Finally, we find it valuable to compare latency to a theoretical lower bound, since user routes to the best unicast alternative may still be inflated.

To measure inflation for the root DNS, we look at how recursive resolvers are directed to sites. DITL captures are a rich source of data because they provide us with a global view of which recursives access which locations for all but a small subset of root DNS sites (§2.1). Our geographic inflation analysis covers 10 of the 13 root letters, omitting G which does not provide data, H which does not provide per-site data and I, where anonymization prevents analysis. (B root's data is anonymized, but preserves /24s, allowing our analysis. We use H root data in later analysis in §4.3).

We calculate latency inflation (as opposed to geographic) for A, M, C, and K root. We do not include latency inflation for other roots due to the deadline, but plan to include the results in the final submission (the results may be interesting,

but will not affect our high-level conclusions). We evaluate latency from the subset of DNS queries that use TCP, using the handshake to capture RTT [55]. We calculate median latency over each ⟨`root, resolver /24, anycast site`⟩ for which we have at least 10 measurements, providing us measurements representing 25% of DITL query volume to these roots, and 98% of queries from resolvers who issued at least one TCP query. Our coverage with this method includes 222 countries and 14400 ASes (Atlas covers about 3700 ASes as of January 2021).

Figure 2a and Figure 2b depict two ways of measuring inflation for users in the DITL∩ACDN data set (§2.1): geographic inflation and latency inflation. Geographic inflation measures, on a high level, how users are routed to sites compared to selection of an optimal line-of-site front-end. On the other hand, latency inflation uses measured latencies to determine inflation, so it reflects constraints due to physical rights-of-way and connectivity, and peering choices.

3.1 Methodology

To calculate geographic inflation, we first geolocate all recursives in DITL∩ACDN data set using MaxMind [6], following prior methodology which affirmed MaxMind to be suitably accurate for geolocating recursive resolvers in order to assess inflation [49]. 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 sites in this letter deployment. $GI_j(R)$ is an approximation of the inflation one would expect to see 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 A.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 how inflation compares among root deployments.

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

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

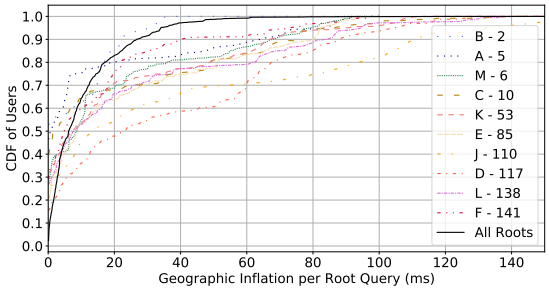
where $l(R, j_i)$ is the median latency of recursive R towards root site j_i and the other variables are as in Equation (1). Prior work notes that routes rarely achieve a latency of less than the great circle distance between the endpoints divided by $\frac{2}{3}$ the speed of light in fiber [45], so we use this to approximately lower bound the best latency recursives could achieve. Latency inflation is therefore a measure of potential performance improvement users could see due to changes in routing or expanding the physical Internet (*i.e.,* laying fiber).

3.2 Results

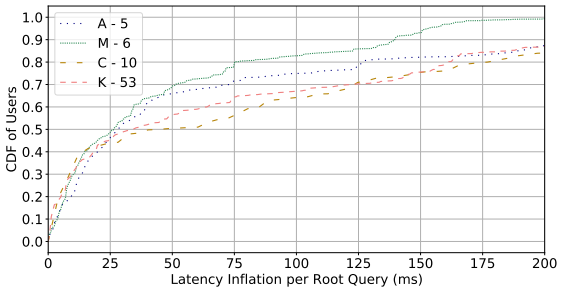
Figure 2a demonstrates that the likelihood of a root DNS query experiencing any inflation (y-axis intercept) roughly grows with deployment size, expanding on results in prior work which presented an orthogonal, aggregated view [49]. The `All Roots` line captures the queries from each recursive to all roots, accounting for how 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 15% of users are likely to be inflated by more than 2,000 km (20 ms).

Figure 2b shows that queries to these roots see frequent latency inflation, with only 40% of users seeing less than 30 ms of latency inflation, and, for all roots we measured, 20% of users see greater than 100 ms. Compared to geographic inflation, latency inflation is larger across the board (as it must be since latency inflation considers factors in addition to physical distance) and can be particularly larger in the tail. For example, at the 95^{th} percentile `C` root has 240 ms of latency inflation but only 70 ms of geographic inflation. 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 [49] (although the comparison to prior work is not perfect since what was measured was slightly 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 [33, 63].

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



(a)



(b)

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

4 ROOT DNS LATENCY AND INFLATION DO NOT MATTER

With a richer understanding of inflation in the root DNS, one might wonder why inflation in the root DNS is so large given root DNS' importance in the Internet. We now show that this 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 crowd-sourced caching 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 [20] or largely replaced by preemptive caching in recursives [47]. 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 so much. The SIGCOMM 2018 paper "Internet Anycast: Performance, Problems, & Potential" has drawn attention to the fact that anycast can inflate latency to anycasted IP addresses by hundreds of milliseconds [49]. 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.

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 C). Hence, caching may not be working as intended.

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 system, 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 [33, 49, 56, 63].

Therefore, precisely determining how root DNS latency affects users would require global, OS-level control to select recursives and view in-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 winter 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 Does Not Matter

Local Perspective: To obtain precise measures of how users interact with the root DNS, we look at settings where we

can see users generate queries to the roots. We use packet captures of a recursive resolver at a research lab (§2.1) to observe how root DNS queries are amortized over a small user population. Since this perspective does not allow us to relate root queries to user experience, we also measure from two authors' computers to observe how an individual user interacts with the root servers (with no crowd-sourced caching). Data from two users is limited, which is a reflection of the challenges we identified in Section 4.2. However, these experiments offer us *precise* measures of how these authors interact with root DNS, supplementing the global-scale data used for most of the paper.

Using traces gathered at the research lab, we calculate the number of queries to any root server as a fraction of user requests to the recursive resolver. We refer to this metric as 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 recursive 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 crowd-sourced caching. Appendix B shows the minimal impact root DNS latency has on users of the research lab, and a CDF of DNS latency experienced by users at the research lab.

Since the measurements at the research lab 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 the research lab. The larger cache miss rate makes sense, given the local users do not benefit from crowd-sourced caching. 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 crowd-sourced DNS caching. In general,

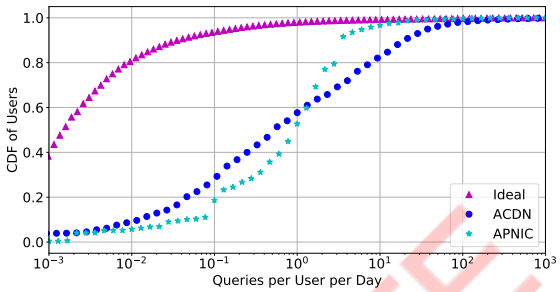


Figure 3: A CDF of the number of queries each user executes to the roots per day. The ACDN and APNIC lines represent different user-count data sets. The Ideal line presents an idealized assumption about recursive query behavior. Most users wait for less than one query to the roots per day, regardless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

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 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 behind 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 ACDN 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. This quotient (representing daily queries per user) is then weighted by user count and the resulting CDF is calculated. 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. As we include nearly every root query captured across the root servers, Figure 3 provides a truly global view of how users incur latency due to the root DNS.

The two lines ACDN and APNIC correspond to amortizing DITL queries over ACDN 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 (ACDN 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. That is, users *rarely* interact with the root DNS executing about one query per day at the median.

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 ACDN user counts for Ideal). The resulting hypothetical median daily query count of .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 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? Moreover, can anycast perform well in systems where latency does matter? To answer these questions, we turn to a new system using anycast to serve latency-sensitive content – ACDN.

5 LATENCY MATTERS FOR CDNS

To establish that anycast CDNs differ in desired properties from the root DNS, we first demonstrate that latency (and hence inflation) *does* matter for users of ACDN when fetching web content, and this is 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 ACDN (§5.2), we first must estimate the number of RTTs required to load a typical web page hosted by ACDN. 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. A lower bound on the number of RTTs to load pages 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 ACDN web pages using Selenium (a headless web browser), finding that 10 RTTs is a reasonable estimate.

ACDN latency occurs when a user downloads 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

$$N = \lceil log_2 \frac{D}{W} \rceil \tag{3}$$

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 [29, 38]. Although W is set by the server, ACDN and a majority of web pages [60] 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 ACDN 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 ACDN'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 ACDN, twenty times for each page. We choose popular ACDN pages with dynamic content suggested to us by an operator of ACDN. We use

Selenium and Chrome to open web pages and use Tshark [15] 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 [16]. 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 (3), and add a final two RTTs for TCP and TLS handshakes. We find only a few percent of ACDN web pages are loaded within 10 RTTs, and 90% of all ACDN page loads are loaded within 20 RTTs, so 10 RTTs is a reasonable lower bound.

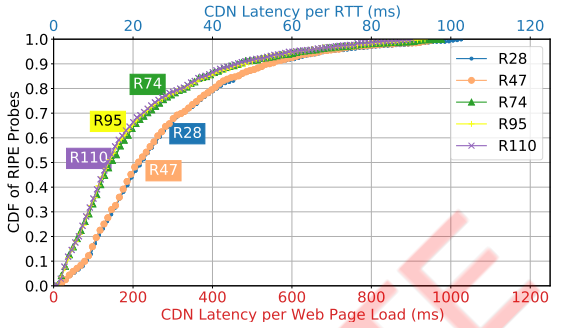
5.2 ACDN User Latency

We now measure how users are impacted by latency of ACDN, representing the study of anycast CDN latency with the best coverage to date. Using measurements from RIPE Atlas probes, we demonstrate that ACDN latency results in significant delay to users for page loads from ACDN. Using both client-side measurements and server-side logs, we also show that latency usually decreases with more sites. Consequently, ACDN has a major incentive to limit inflation seen by users, and investments in more anycast sites positively affect user experience much more in the case of ACDN than in the roots.

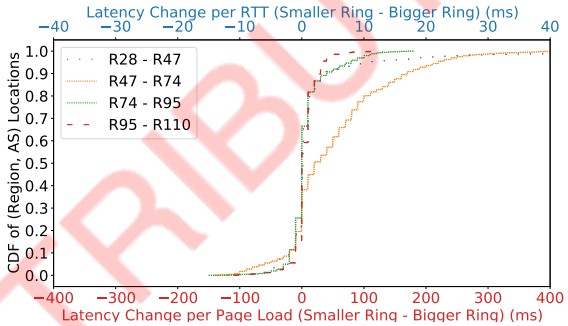
ACDN has groups of sites called *rings* (§2.2). Each larger ring adds some sites to those of the smaller ring. Rings provide some caching and use split TCP connections, serving as front-ends to cloud services for content. Since each ring provides an IP anycast CDN, we report results for each of the rings individually. Different ring sizes reflect some of the benefit of additional anycast locations, but a user's traffic usually ingresses to ACDN's network at the same PoP regardless of ring, since all routers announce all rings.

Users experience latency from ACDN as they retrieve web objects (*e.g.,* web pages or supporting data) hosted by ACDN. Hence, in order to assess how users of ACDN 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), since we can not share absolute latencies from ACDN measurements. Although RIPE Atlas has limited coverage [22], we have ground truth latencies from all ⟨`region, AS`⟩ to all rings. We observed that the distribution of RIPE Atlas probe latencies is overall somewhat lower than that of ACDN's users globally (not shown in figure), so Figure 4a likely underestimates the latency users typically experience.



(a)



(b)

Figure 4: RTTs and latencies per web page load from RIPE probes to ACDN rings (4a), and change in median latency for ACDN 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 ACDN has an incentive to reduce inflation.

Users can experience up to 1,000 ms in 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 ACDN factors into user experience, and so ACDN has an incentive to decrease latency for users. The difference in median latency per page load experienced by RIPE probes 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 (not shown due to length restrictions), but such reductions in latency do not benefit users (§4).

Latency benefits with more sites are not uniform, 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 ACDN users with a geographically close front-end over R47 (Fig. 1).

To show how adding front ends tends to help individual ⟨region, AS⟩ (in addition to aggregate performance), Figure 4b shows change in latency for ⟨region, AS⟩ locations as ring size increases, calculated using ACDN 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, although a small fraction of users see small increases in latency when moving to larger rings.

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

6 ANYCAST INFLATION CAN BE SMALL

Latency matters for users of ACDN, and so ACDN has a major incentive to reduce inflation. We next investigate whether this incentive 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 taken over billions of users in hundreds of countries and thousands of ASes.

To measure anycast inflation for ACDN we use geographic information and server-side measurements (§2.2). Server-side logs are valuable for calculating latency inflation, as these measurements give us a global view of which clients hit which front-end sites and the latencies they achieved. Latency is measured via server-side logging of TCP round-trip times. We also use ACDN 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⟩ are assigned the mean location of users in the ⟨region, AS⟩. Anycast inflation results in extra latency for every packet (and corresponding ACK) exchanged between a client and an anycast 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).

ACDN users usually experience no geographic inflation (Fig. 5a, y-axis intercepts), and users are seldom directed to far-away nodes, with 85% of users experiencing less than 10 ms (1,000 km) of geographic inflation per RTT for all ACDN rings. Conversely, 97% of root DNS users experience some inflation, and 35% of users see inflation more than 10 ms (1,000 km) per RTT. The fact that inflation is much larger and more prevalent than ACDN inflation per RTT than in the roots (at every percentile) suggests ACDN optimizes its deployment to control it (§7).

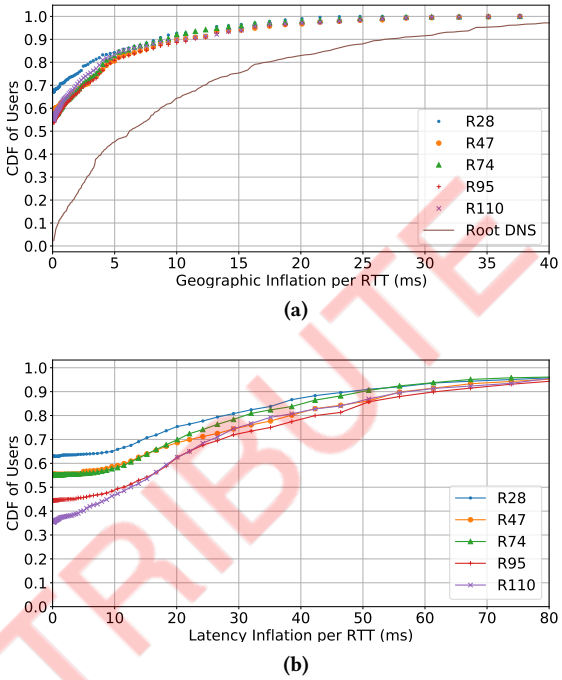
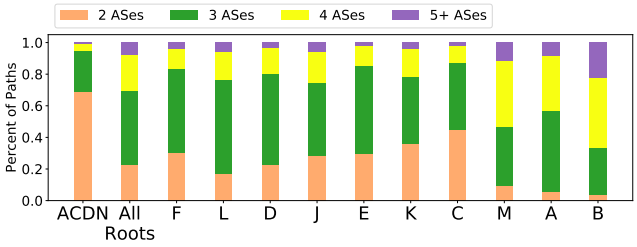


Figure 5: Inflation measured using geographic information (5a) and ACDN server side logs (5b). Inflation is more prevalent for larger deployments but is still small for most users.

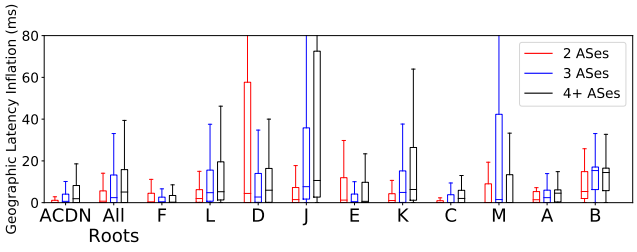
We next calculate latency inflation for each ring. We calculate median latencies over user populations within a ⟨region, AS⟩ hitting a front-end in a given ring, the assumption being that measurements from some users in a ⟨region, AS⟩ hitting the same site are representative of all users in that ⟨region, AS⟩ hitting that site (we study a widely used CDN). More than 83% of such medians were taken over more than 500 measurements, so our observations should be robust.

We compute latency inflation as in Equation (2). There is greater latency inflation as the number of front-ends grows (Fig. 5a) – for example, in R28, only approximately 35% of users experience any inflation, while in R110, 65% of users experience some inflation. However, the CDN is able to keep *all* latency inflation below 30 ms for 70% of users in all rings and below 60 ms for 90% of users. For roots, only 40% of users see less than 30 ms of inflation (Fig. 2b).

Compared to Figure 5a, Figure 5b demonstrates that looking at latency inflation shows that more users experience a small amount of latency inflation than they do geographical inflation ("flatter" left-part of latency inflation curves). Hence, even though clients generally visit close sites (Fig. 5a), there is still room for latency optimization, which is an active area of research [42, 46, 75].



(a)



(b)

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 latency inflation (6b). ACDN is closely connected to many eyeball ASes, and this connectivity correlates with lower inflation. We group paths towards roots and ACDN by ⟨region, AS⟩, except for 'All Roots' which groups paths by ⟨region, AS, root⟩.

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 ACDN by looking at path lengths (§7.1) and presenting reasons behind the expansion of both root DNS and CDNs (§7.2).

7.1 CDNs Have 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. ACDN deploys state-of-the-art network routing automation [62, 73], a global SDN WAN [39, 41], and expensive peering agreements when they make economic sense or help user experience. These optimizations and strategies result in short, low latency routes between users and ACDN.

We can capture some of these engineering efforts by measuring how ACDN connects to users. CDNs peer widely with end-user networks and so have direct paths to many users [52, 71]. With fewer BGP decision points, paths are often less inflated [64]. This intuition motivates the following investigation of AS path lengths towards roots and ACDN 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 ACDN's optimizations.

To quantify differences in AS path length between ACDN and roots, Figure 6a shows AS path lengths to roots and ACDN 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 7200 RIPE Atlas probes in 158 countries and 2400 ASes. Although RIPE Atlas probes do not have representative coverage [22], it is the best publicly available system, and we are only interested in qualitative, comparative conclusions.

Lengths towards ACDN 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).[4] We perform IP to AS mapping using Team Cymru [5], 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 [7]. We group paths by ⟨region, AS⟩, except for 'All Roots', for which we group paths by ⟨region, AS, root⟩. Specifically, we map RIPE Atlas probes to ⟨region, AS⟩, and the AS path length from a ⟨region, AS⟩ to a destination is a combination of all path lengths measured from RIPE Atlas probes in that ⟨region, AS⟩ to that destination, with each RIPE Atlas probe given equal weight (*i.e.,* each path is given unit weight).

Figure 6a shows shorter paths to ACDN than to the roots. (Weighting by traffic volumes yielded similar results.) 69% of all ACDN paths only traverse two ASes (direct from RIPE Atlas probe AS to destination AS), and only 5% of ACDN paths 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 latency inflation. For the inflations towards destinations in Figure 6b, we use the geographic latency inflation associated with that ⟨region, AS⟩ 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⟩. Figure 6b demonstrates that paths that traverse fewer ASes tend to be inflated less. Except D and E roots, median inflation for AS paths of length two is less than median inflation for longer paths.

[4] 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. When the 2020 DITL captures become available, we plan to include the updated results.

Overall, our results demonstrate that shorter paths tend to have less inflation, users have shorter paths to ACDN than towards the roots, and ACDN tends to have less inflation across path lengths. We believe these observations are a result of strategic business investments that ACDN puts toward peering and optimizing its routing and infrastructure. In addition to shorter AS paths generally being less inflated [64], ACDN's direct paths in particular sidestep the challenges of BGP by aligning the best performing paths with the BGP decision process [30]. 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 ACDN will only exist during a route leak/hijack). Among the multiple direct paths to ACDN that a router may learn when its AS connects to ACDN in different locations, the decision will usually fall to lowest IGP cost, choosing the nearest egress into ACDN. ACDN 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 [64]. At smaller ring sizes, ACDN 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 Differing Incentives Lead to Different Investments and Outcomes

Given these differences in anycast systems, we return to our question: why are anycast networks growing in size? To answer this question, we reached out informally to operators of both root DNS and ACDN, and we summarize our discussions.

Over the past 5 years the number of root DNS sites has more than doubled from 516 to 1367, steadily increasing. Once a DNS service is beyond a dozen sites or so, new sites are deployed primarily to increase DNS resilience and address tail latency. Growth stems in part from root server operators with open hosting policies [2, 13, 17] – almost any AS can volunteer to host a new instance at relatively low cost. ASes or governments might want to host a site to provide maximum availability to users or citizens. With such decentralized deployment (in part by design to prompt resilience), even if latency optimization were a goal, coordinated optimization of root DNS latency is difficult.

By contrast, the 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). 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 ACDN has more than doubled in the past five years.

8 RELATED WORK

Root DNS Anycast. Prior work looks at anycast's ability to load balance and defend against DDoS attacks [56, 61]; we do not consider anycast's performance in this context. Many prior studies look at latency and inflation performance in the root DNS [25, 49, 50, 61, 63], however none relate performance to user experience, nor do they compare performance to other systems. Other prior work discussed how deploying anycast in an ad-hoc manner can lead to poor performance and load balancing and is an early study of inflation in the root DNS [25]. 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 [24], at least over the duration of DITL.

CDN Anycast. Some CDNs use IP anycast [26, 31, 36, 59, 68]. Some prior work looked at inflation in CDNs [26], finding it to be similarly low. Our work presents a much larger study of latency and inflation, lending confidence to the results; 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 [52]. More recent work has investigated how to diagnose and improve anycast performance through measurements in production systems [69]. Our work characterizes, rather than changes, anycast CDN performance.

Recursive Resolvers and the Benefits of Caching. Prior work has looked at statistics and latency implications of local resolvers [28, 43]. 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 [37, 48, 67, 74]; we present additional pathological behavior of a popular recursive in Appendix C.

Web Performance. Many studies characterize web performance and consider DNS's role in a page load [11, 23, 66], 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 [21].

9 CONCLUSION

While anycast performance is interesting in its own right, prior studies have drawn conclusions primarily from anycast for root DNS [49]. We have shown that anycast operates quite differently in CDNs, with much less inflation. We suggest that this difference stems from the impact the deployment has on what users perceive. Our work shows the importance of considering multiple subjects in measurement studies and suggests why anycast continues to see wide and growing deployment.

REFERENCES

[1] 2018. DNS-OARC. dns-oarc.net/oarc/data/ditl/2018

[2] 2018. Hosting a K-root node. ripe.net/analyse/dns/k-root/hosting-a-k-root-node

[3] 2019. The Top 1,000 Sites on the Internet. gtmetrix.com/top1000.html

[4] 2020. developers.google.com/speed/public-dns

[5] 2020. team-cymru.com/community-services/ip-asn-mapping/

[6] 2020. maxmind.com/en/geoip2-databases

[7] 2020. caida.org/data/as-organizations/

[8] 2020. Amazon Route 53 FAQs. aws.amazon.com/route53/faqs/

[9] 2020. Data center Locations. opendns.com/data-center-locations/

[10] 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

[11] 2020. The HTTP Archive Project. httparchive.org/

[12] 2020. IANA IPv4 Special-Purpose Address Registry. iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

[13] 2020. Packet Clearing House. icannwiki.org/Packet_Clearing_House

[14] 2020. Root Servers. root-servers.org

[15] 2020. Tshark. wireshark.org/docs/man-pages/tshark.html

[16] 2020. Window: load event. developer.mozilla.org/en-US/docs/Web/API/Window/load_event

[17] 2021. FAQ on RIRS Node Hosting. verisign.com/en_US/domain-names/internet-resolution/node-hosting/index.xhtml

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

[19] 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. (2020). icann.org/en/system/files/files/octo-008-en.pdf

[20] Mark Allman. 2019. On Eliminating Root Nameservers from the DNS. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 1–8.

[21] Mark Allman. 2020. Putting DNS in Context. In *ACM SIGCOMM Internet Measurement Conference*.

[22] 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 *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 79–88.

[23] Alemnew Sheferaw Asrese, Pasi Sarolahti, Magnus Boye, and Jorg Ott. 2016. WePR: a tool for automated web performance measurement. In *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 1–6.

[24] Hitesh Ballani and Paul Francis. 2005. Towards a global IP anycast service. *ACM SIGCOMM Computer Communication Review* 35, 4 (2005), 301–312.

[25] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. 2006. A measurement-based deployment proposal for IP anycast. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 231–244.

[26] 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*. ACM, 531–537.

[27] 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} 18). 501–517.

[28] Thomas Callahan, Mark Allman, and Michael Rabinovich. 2013. On modern DNS behavior and properties. *ACM SIGCOMM Computer Communication Review* 43, 3 (2013), 7–15.

[29] Neal Cardwell, Stefan Savage, and Tom Anderson. 2000. Modelling TCP Latency. In *Proceedings of the IEEE Infocom* (johnh: pafiles). IEEE, Tel-Aviv, Israel, to appear. http://www.cs.washington.edu/homes/cardwell/papers/infocom2000tcp.pdf

[30] 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*. 523–529.

[31] 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*. 1–13.

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

[33] Lorenzo Colitti, Erik Romijn, Henk Uijterwaal, and Andrei Robachevsky. 2006. Evaluating the effects of anycast on DNS root name servers. *RIPE document RIPE-393* 6 (2006).

[34] 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* 24, 1 (2014), 43–57.

[35] Manaf Gharaibeh, Han Zhang, Christos Papadopoulos, and John Heidemann. 2016. Assessing co-locality of IP blocks. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 503–508.

[36] Danilo Giordano, Danilo Cicalese, Alessandro Finamore, Marco Mellia, Maurizio M Munafò, Diana Zeaiter Joumblatt, and Dario Rossi. 2016. A First Characterization of Anycast Traffic from Passive Traces.. In *TMA*.

[37] Wes Hardaker. 2020. Whats in a Name. blog.apnic.net/2020/04/13/whats-in-a-name/

[38] John Heidemann, Katia Obraczka, and Joe Touch. 1997. Modelling the Performance of HTTP Over Several Transport Protocols. *ACM/IEEE Transactions on Networking* 5, 5 (Oct. 1997), 616–630. https://www.isi.edu/%7ejohnh/PAPERS/Heidemann96a.html

[39] 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 ACM SIGCOMM 2013 conference on SIGCOMM*. 15–26.

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

[41] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. 2013. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 3–14.

[42] 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 ACM Special Interest Group on Data Communication*. 104–116.

[43] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on networking* 10, 5 (2002), 589–603.

[44] Dina Katabi and John Wroclawski. 2000. A framework for global IP-anycast (GIA). *ACM SIGCOMM Computer Communication Review* 30, 4 (2000), 3–15.

[45] 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 6th ACM SIGCOMM conference on Internet measurement*. 71–84.

[46] 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 9th ACM SIGCOMM conference on Internet measurement*. 190–201.

[47] W. Kumari and P. Hoffman. 2020. *Running a Root Server Local to a Resolver*. RFC 8806. Internet Request For Comments. https://doi.org/10.17487/RFC8806

[48] Matthew Lentz, Dave Levin, Jason Castonguay, Neil Spring, and Bobby Bhattacharjee. 2013. D-mystifying the D-root Address Change. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 57–62.

[49] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet anycast: performance, problems, & potential.. In *SIGCOMM*. 59–73.

[50] 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*. Springer, 145–154.

[51] Zhuoqing Morley Mao, Charles D 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, General Track*. 229–242.

[52] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. 2019. Taming Anycast in the Wild Internet. In *Proceedings of the Internet Measurement Conference*. 165–178.

[53] Christopher Metz. 2002. IP anycast point-to-(any) point communication. *IEEE Internet Computing* 6, 2 (2002), 94–98.

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

[55] GC Moura, John Heidemann, Wes Hardaker, Jeroen Bulten, Joao Ceron, and Cristian Hesselman. 2020. Old but gold: Prospecting tcp to engineer dns anycast (extended). *Tech. Rep. ISI-TR-740, USC/Information Sciences Institute, Tech. Rep.* (2020).

[56] Giovane Moura, Ricardo de O 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*. ACM, 255–270.

[57] Moritz Müller, Giovane C. M. Moura, Ricardo de O. Schmidt, and John Heidemann. 2017. Recursives in the Wild: Engineering Authoritative DNS Servers. In *Proceedings of the ACM Internet Measurement Conference* (johnh: pafile). ACM, London, UK, 489–495. https://doi.org/10.1145/3131365.3131366

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

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

[60] Jan Rüth, Christian Bormann, and Oliver Hohlfeld. 2017. Large-Scale Scanning of TCP's Initial Window. In *Proceedings of the 2017 Internet Measurement Conference*.

[61] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. 2006. On the use of anycast in DNS. In *Proceedings of 15th International Conference on Computer Communications and Networks*. IEEE, 71–78.

[62] 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 Conference of the ACM Special Interest Group on Data Communication*. 418–431.

[63] 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*. Springer, 188–200.

[64] 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*. 113–124.

[65] RIPE NCC Staff. 2015. Ripe atlas: A global internet measurement network. *Internet Protocol Journal* 18, 3 (2015).

[66] 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/international conference on Measurement and modeling of computer systems*. 383–384.

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

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

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

[70] Lan Wei and John Heidemann. 2017. Does anycast hang up on you?. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–9.

[71] 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 Conference of the ACM Special Interest Group on Data Communication*. 206–220.

[72] Jing'an Xue, Weizhen Dang, Haibo Wang, Jilong Wang, and Hui Wang. 2019. Evaluating performance and inefficient routing of an anycast CDN. In *Proceedings of the International Symposium on Quality of Service*. ACM, 14.

[73] 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 Conference of the ACM Special Interest Group on Data Communication*. 432–445.

[74] Yingdi Yu, Duane Wessels, Matt Larson, and Lixia Zhang. 2012. Authority server selection in DNS caching resolvers. *ACM SIGCOMM Computer Communication Review* 42, 2 (2012), 80–86.

[75] 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* 9, 3 (2012), 333–345.

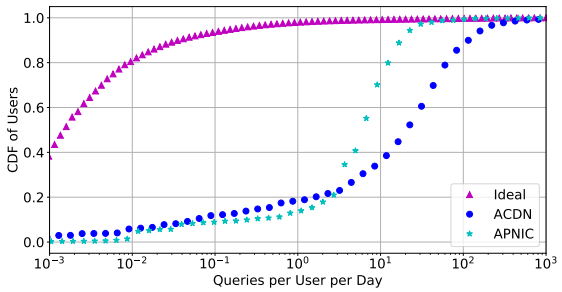


Figure 7: Daily queries by users to the root DNS, calculated by amortizing root DNS requests over user populations, when including or excluding queries for invalid TLDs. Counting invalid queries drastically increases median daily query counts to 22 (ACDN), a 20-fold increase, or to 6 (APNIC), a 6-fold increase, depending on which user data we use.

A 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 preprocessing steps, and analyze the effects of these assumptions on our results.

A.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 [19]. Researchers at USC have found that more than 90% of single-label queries at the root match the Chromium captive-portal pattern [37]. We remove captive-portal detection queries from user latency because it occurs on browser startup and network reconnect, not during regular browsing, and it 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 [34]. 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 7 shows daily user latencies due to root DNS resolution when we include requests for invalid TLDs and PTR records in daily query volumes. Using ACDN 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 7 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.

A.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 users of ACDN who use recursives (§2.1). Naturally recursives used by users of ACDN and recursives seen in DITL do not overlap perfectly. To increase the representativeness of our analysis, we aggregate ACDN user counts and DITL query volumes by resolver /24, and join the two datasets on /24 to create the DITL∩ACDN data set. 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

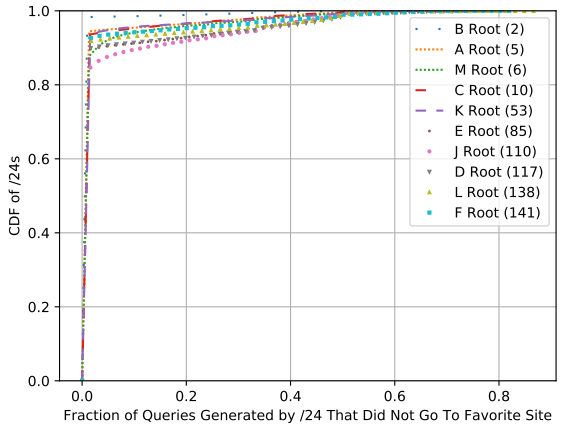


Figure 9: Fractions of queries generated by /24s that do not hit the most popular site for each /24 and for each root letter in question. For all root letters, more than 80% of /24s have all queries visit the most popular site, suggesting queries from the same /24 are usually routed similarly.

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

Table 1: Statistics displaying the extent to which the recursives of users in ACDN 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%.

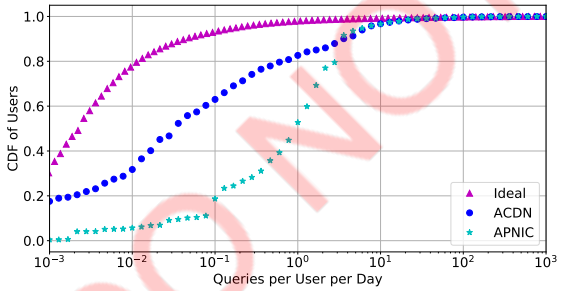


Figure 8: A CDF of the number of queries users of ACDN experience due to root DNS resolution, per day, without joining recursives in DITL with recursives seen by ACDN by /24 (ACDN). This unrepresentative analysis yields an estimate of daily user queries far lower than in Section 4.3.

now justify this decision and discuss the implications of this preprocessing step on the results presented in Section 4.3.

In Table 1 we summarize the extent to which the recursives seen by ACDN 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 uniformly by tens of percents and in certain cases by up to 64%.

As an analogy to Figure 3, in Figure 8 we show the number of queries each user of ACDN executes to the roots per day *without* aggregating query and user statistics by /24 (ACDN). We also show APNIC as in Figure 3 for comparison, even though APNIC is not affected by /24 volume aggregation. Users of ACDN only send .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 1).

Table 1 and Figure 8, demonstrate that the decision to aggregate statistics and join DITL captures with ACDN 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 [70], 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 9 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. Specifically, 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

$$f^k = 1 - \sum_i \frac{q_{ij_F}^k}{Q^k} \quad (4)$$

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.[5]

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

[5]We do not include /24s that had only one IP from the /24 visit the root letter in question.

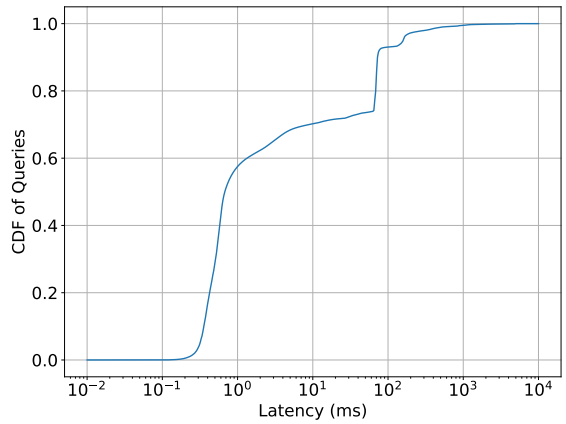


Figure 10: CDF of user DNS query latencies seen at a recursive resolve at ISI, over the course of one year. Latencies are measured from the timestamp when the recursive resolver receives a client query to the timestamp when the recursive sends a response to that client query. The sub-millisecond latency for more than half of queries suggests most queries to this recursive are served by the local cache.

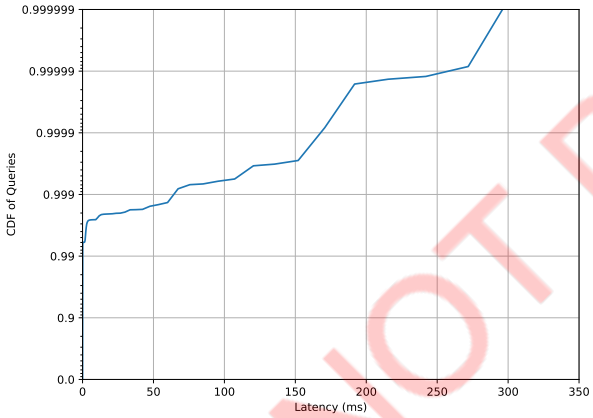


Figure 11: Root DNS latency for queries made by users of the research lab recursive resolver during 2018. This plot demonstrates the benefits of crowd-sourced caching and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than .1% incur latencies greater than 100 ms**. User queries that did not generate a query to a root server were given a latency of 0.**

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 9 provides evidence that recursives 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 in all roots are routed to the same site (not shown).

B LATENCY MEASUREMENTS AT A RECURSIVE RESOLVER

To obtain a local perspective of how users experience root DNS latency, we use packet traces from a research lab in a university in the United States. 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.

Figure 10 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. 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 [28]. 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 10. To visualize just how small this fraction is, Figure 11 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 11 demonstrates the benefits of crowd-sourced caching and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than .1% incur latencies greater than 100 ms.

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

Table 2: Pattern example of the redundant root DNS requests. The last five requests to J root are redundant which may be caused by an unanswered request in step 4.

C 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 [34].

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 is in BIND's cache. We use BIND version 9.10.3 and 9.16.1. Because 9.16.1 is one of the newest releases and 9.10.3 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 [3] using Selenium and headless Chrome, While loading web pages, we collect network packets on port 53 using Tshark [15].

For these page loads, we observe 69215 DNS A & AAAA-type requests generated by the recursive resolver. 3137 of these requests are sent to root servers and 2950 of them 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 2.

In Table 2, 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 doesn't 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 2, 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 a research lab in a university in the United States. To keep consistent with the other analysis we do on this data set (§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.

Now we know that a large number of unnecessary DNS requests to the roots are redundant and are generated in BIND v9.16.1 and v9.10.3. Software behaviors like the one 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.

D ANYCAST CDN PERFORMANCE VISUALIZATION

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 12 we show front-ends in R110, and associated performance users see to R110 in each region. Transparent circles 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 see to R110 – red indicates higher latency while green indicates lower latency. Latency generally gets lower,

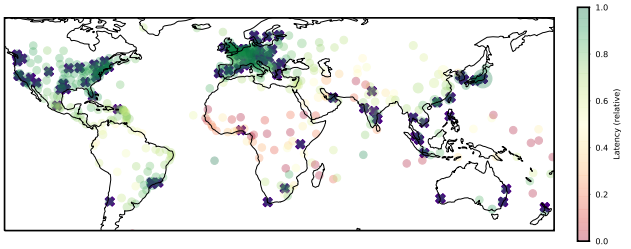


Figure 12: A visualization of front-ends in R110 (purple Xs), and user populations (transparent circles). User populations are colored according to the relative latency they see, and have size proportional to user population. Red corresponds to high latency and green corresponds to low latency. Latency generally gets lower, the closer users are to a front-end, and front-ends are concentrated around large user populations.

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.