Anycast in Context: A Tale of Two Systems

ABSTRACT

Anycast is widely used today to provide many types of content, including web-content from CDNs and services such as DNS. Research, often based on analysis of root DNS, has suggested that path inflation is a problem for anycast, with users often routed to suboptimal sites. However, this understanding conflicts with practice, since many large latency-sensitive services use anycast. We reassess conclusions about path inflation with experiments using global traces from a large anycast CDN and root DNS traffic. We first show that, while latency is important to CDNs, it is largely irrelevant for root DNS – even with imperfect anycast routing, most users experience less than 15 ms per day from root DNS because caching is so effective. Second, we show users experience lots of inflation when they query the root DNS; however, even with that inflation, performance is near optimal from a user quality-of-experience perspective. In an anycast CDN serving latency-sensitive content, as low as 35% of users see any inflation, and the inflation users do see is significantly less than in the root DNS. We provide evidence that the CDN is able to keep inflation low due to extensive peering and engineering effort. While path inflation happens, it is not a problem that hinders users of the DNS root today, and it does not severely impact CDN performance. Overall, our results highlight the importance of considering a technique like anycast in the context of the system that uses it, rather than evaluating it independant of a particular use case.

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) [3, 7, 9, 12, 49] and Content Delivery Network (CDN) [17, 21, 59] systems today, in part because of its ability to improve latency to clients and decrease load on each anycast server [37, 45, 48].

However, studies have argued that anycast provides sub-optimal performance for some users, compared to the lowest latency one could achieve given deployed sites [21, 40, 44, 51, 62]. Results have suggested that anycast routing is ineffective, failing to direct users to the lowest latency site [40, 41, 51]. However, the degree to which this inefficiency affects end-users and generalizes to other networks has not been studied.

To understand the actual impact of anycast efficiency, we take a step back and evaluate anycast as a technique used by systems, rather than as a system in-of-itself. Performance implications to users depend on the particulars of the anycast deployment, how the anycast deployment is used within the system, and how users interact with the system. To study anycast from this perspective, we analyze anycast's role within two different, real-world systems: the root DNS and a large anycast CDN, chosen because they have related, yet distinctive, goals, they are key components of the Internet, and they are two of the dominant, most studied use cases.

Data from the root DNS (*e.g.,* packet traces) are readily available [12] and are featured in existing anycast studies [27, 28, 40, 44, 47]. 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 [1], 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 to each ring, giving us a complete view of `ACDN` performance.

With these measurements, we present the largest study of anycast latency and inflation to date. To understand how anycast affects performance, we first look at both anycast latency and user-perceived latency for these systems. We find that root DNS anycast latency can be variable, ranging from a few to several hundred milliseconds. However, these differences are hardly perceived by users – using global-scale traces we show that root DNS queries account for a few milliseconds of wait time per day, or a percent of a page load (§4.1). Delay is minimal due to caching of root DNS records with long TTLs at recursive resolvers. Hence, we show that there is no value to lowering root DNS latency, since root DNS latency is already near optimal from a user-quality-of-experience perspective. In contrast, greater latency to users using websites backed by `ACDN` can result in orders of magnitude more latency over a day than in the root DNS (§4.2). Hence larger CDN deployment sizes

can reduce page load times by over a hundred milliseconds, but provide little latency benefit in the root DNS setting. Collectively, these results suggest that organizations managing systems that use anycast have different incentives and strategies when deploying anycast, which should be taken into account when analyzing anycast inflation and efficiency (§4.3).

With these latency results, we reassess prior understanding of anycast inflation, finding that inflation is small in systems where inflation hurts users. 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. We also show that inflation roughly increases with the number of anycast sites (§5.1). We compare these inflation results with `ACDN` inflation and find that inflation is kept comparatively small for `ACDN`. We also show that global latency per page load can decrease by hundreds of milliseconds when adding more sites, even though adding sites increases inflation (§5.2). Moreover, regardless of deployment size, the inflation per RTT for `ACDN` is about half that of the root deployments. 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, unlike the root servers, which have little incentive to keep latency low, `ACDN` is able to control inflation through extensive peering and engineering investment (§5.3). Hence, by considering performance implications to users, we have shown that anycast is a valid technique to use in Internet systems that distribute latency-sensitive content to users.

2 MOTIVATION

Global Internet services often focus on latency as a performance measure because it has been closely tied to user experience and revenue [42, 56]. IP anycast is one approach to reducing latency by routing to one of many geographically diverse servers, known as anycast sites, that provide the same service with a single IP address. Anycast's operational simplicity makes it attractive – global traffic management is handled by the network. Studies of anycast have suggested that this simplicity comes at a cost – inflated latency [20, 21, 27, 28, 40, 51]. We now discuss this conventional wisdom and how we plan to reassess it.

2.1 A Conventional Wisdom

Conversation with experts and practitioners in the Internet measurement community have revealed that the networking community is skeptical of anycast performance. 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 IPs by hundreds of milliseconds [40]. According to this study, anycast has a problem, but could be "fixed" with modifications to BGP community conventions. However, this understanding flies in the face of practice – given anycast's sub-optimal performance, why do several large CDNs (which are latency-sensitive) successfully use anycast today [21, 49, 59]?

2.2 Two Second Looks

We reassess this wisdom by evaluating anycast as a technique used by systems, rather than as a system in-of-itself. Performance implications to users depend on the particulars of the anycast deployment, how the anycast deployment is used within the system, and how users interact with the system. We introduce two important, distinct systems on the Internet that use anycast – the root DNS and CDNs.

2.2.1 The 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 [26, 46]. 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 [12], each with a different anycast deployment with 6 to 252 anycast sites (as of May 2020), run by 12 commercial and non-profit organizations. With only about 1000 names, almost all cacheable for 2 days, recursives need only query the root DNS rarely and results will be shared over many clients. We explore DNS request amortization in Section 4.1. (Amortization is not this simple, since recursives may prefetch records or have software bugs causing unnecessary querying.)

2.2.2 Anycast CDNs. The second system we study is a large anycast CDN (which we call `ACDN`) 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 sites into a logical hierarchy of layers, called rings, that conform to varying degrees of regulatory restrictions, each with its own anycast address. Hence, traffic from

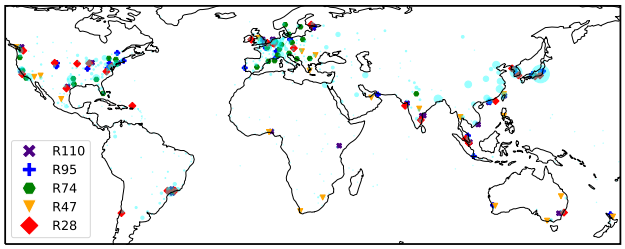


Figure 1: The hierarchical ring structure, and its global presence. User populations are shown as circles, with the radius of the circle proportional to the number of users in that region, demonstrating that ACDN has deployed front-ends in areas of user concentration.

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.

Clients of CDNs are associated with servers either by anycast, by selection through DNS, or with both methods [21]. ACDN is operated by one organization, with 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. ACDN has a global measurement system, capable of measuring latencies from large populations of users (§3).

Unlike the root DNS, ACDN serves large, commercial objects (web and media), instead of short and public-service DNS. As a result, ACDN operators are motivated to minimize latency to improve customer performance.

2.3 Key Challenges

Our goal is to understand anycast performance, to gain a better understanding of the cost of anycast inflation and large growth in investment in anycast systems. Our new perspective is to consider these two different applications: root DNS and CDNs, and to focus on *user-visible latency*.

ACDN has measurement tools in place to evaluate object latency, but latency for the DNS is much more challenging. 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 (anycast service). These challenges not only motivate the subsequent analyses (§4, §5), but also highlight the shortcomings of prior work which do not capture these subtleties of root DNS latency [27, 28, 40, 47].

Precisely determining how root DNS latency affects users would require global, OS-level control to select recursives and view in-OS DNS cache, 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 fall 2020, only Google might have this data, and assembling it would be daunting.

2.4 A New Understanding

These challenges prompt us to develop new methodologies that provide different tradeoffs in precision and coverage to study how anycast latency translates to end users, particular for root DNS. Our methodologies for studying root DNS latency suggest an intuitive result that nonetheless has not been articulated in studies of anycast latency based on root traces: root DNS latency *hardly matters* for users due to caching (§4.1).

This result about root DNS latency, paired with a detailed analysis of inflation in ACDN, allows us to put results from prior work in perspective. Even though root inflation is large, users rarely experience it, making average inflation quite small. In contrast, users frequently interact with anycast CDNs, but inflation rarely occurs. When inflation does occur, it is much smaller than in the root DNS. These inflation results make sense, given the economic incentives of the organizations running ACDN and the root DNS. 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.

3 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 ACDN data is proprietary, gathered from ACDN's internal measurement system. We supplement these datasets with measurements from RIPE Atlas [55] which has more than 11k active measurement probes in hundreds of countries.

3.1 Root DNS Data and Methodology

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 anomalies, like measurement experiments, in the period we use.

To obtain a global view of root DNS use and how these queries translate to user latency, we use the 2018 DITL captures, archived by DNS-OARC [1] (2018 is the most recent available as of September 2020). DITL occurs annually, with each event including data from most root servers. The 2018 DITL took place 2018-04-10 to -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 information on addresses more specific than /24, so we use all data from B root.

Since we aim to understand 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 preprocessing 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 captive portal detection from Chromium-based browsers [15, 31, 58], and so involve machine startup and not browsing latency. Prior work suggests the remainder are generated by other malfunctioning, automated software [29]. 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 including invalid TLD queries increases our daily root DNS latency estimates by an order of magnitude, and provide more justification for this step.

Sources of DNS queries in DITL are typically recursive resolvers, so the captures alone provide little information about how many users use each recursive resolver, or how many DNS queries each user makes. To estimate per-user latency, we augment these traces with the approximate number of users of each recursive, gathered in 2019 (the oldest user data we have). This user data is from DNS data at `ACDN`, and it counts unique IP addresses as "users". This definition undercounts multiple human users that use a single IP address with Network Address Translation (NAT). `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` [22, 43].

Since we aim to amortize root DNS queries over users, 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 bych /24 prefix.[1] We aggregate by /24s to increase the amount of recursives for which we have user data, noting that many organizations use colocated servers within the same /24 as recursives [3, 8]. Prior work has also found that up to 80% of /24's are collocated [30]. 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 site. For clarity, 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`. We next remove queries from prefixes in private IP space [11] (7% of all queries). Finally, we analyze only IPv4 data and exclude IPv6 traffic (12% of queries) since we do not have IPv6 user data.

In joining root DNS recursives with `ACDN` user data, we obtain precise user count information, yet trade-off some coverage (*i.e.,* the rest of the recursives). In Appendix A.3 we use a different data source of user populations and methodology to amortize DITL queries over populations and arrive at roughly the same conclusions about root DNS latency, lending credence to our overall conclusions about root latency experienced by users.

3.2 `ACDN` Data Sources

To study performance in `ACDN`, we use two major data sources: server-side logs and client-side measurements. Server-side logs (*i.e.,* 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

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

serves them.[2] ACDN determines the location and AS of users using internal databases.

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 about 500 regions in total: 130 in Europe, 60 in Africa, 100 in Asia, 1 in Antarctica, 130 in North America, 40 in South America, and 25 in Oceania.

Client-side measurements come from a measurement system operated by ACDN, similar to those described in the literature [22, 43]. Latency measurements are the time it takes to fetch a small image via HTTP.[3] From this measurement system, ACDN collects latencies of users populations, noting the location and AS of the user. We are able to hold the user population constant for a given ⟨region, AS⟩ across rings, which enables us to remove biases in latency patterns due to differences between enterprise and residential traffic. 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, so we can directly compare latency to those collected by ACDN, to provide latency measurements we can share (latency from ACDN is considered proprietary). In total, we collect 7,000 measurements from 1,000 RIPE probes in more than 500 ASes to augment ACDN latency measurements.

4 LATENCY IMPACT ON END USERS

To understand anycast performance in the context of two systems, we first investigate latency for both the root DNS and ACDN. We quantify the latency that users experience, how latencies between systems compare to each other, and the effects that additional sites have on user latency. Analyzing how users interact with these systems allows us to place path inflation in context of user perceived latency in Section 5.

4.1 Impact of Root DNS Latency on Users

To assess how root DNS performance impacts end users, we take two perspectives: 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 and over browsing sessions, while our global analysis estimates the total time per day users worldwide wait for root DNS resolution. We find that users rarely experience a root DNS query during a page load (local view) and experience no more than a few tens of milliseconds per day waiting for root DNS resolution (global view). Hence, root DNS performance matters little to the user.

	Statistic (Median)	Value
Shared Recursive Resolver Cache	Root Cache Miss Rate	0.5%
Single User Resolvers	Root Cache Miss Rate	1.5%
	Total Root DNS Latency (s)	4.63
	Total DNS Latency (s)	275.9
	Active Browsing Time (s)	10740
	Page Load Time (s)	382.5

Table 1: Statistics gathered from the research lab recursive (top), and from two authors' machines (bottom). Root DNS queries are infrequent, making up a small fraction of total queries from clients.

A Local Perspective of Root DNS Latency

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. These experiments have limited coverage, but provide a precise measure of how often these users interact with the root DNS. We use packet captures of a recursive resolver at a research lab (§3.1) 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 a single user interacts with the root servers (with no crowd-sourced caching).

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.

Table 1 summarizes our findings from all local root DNS experiments, and in particular the median root cache miss rates seen at the research lab. 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 global cache miss rate across 2018 was also 0.5%. The particular cache miss rate may vary depending on user

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

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

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 4 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 this user when loading web pages, even without crowd-sourced DNS caching. In general, we *overestimate* the impact of DNS and root DNS latency since DNS queries can occur as a result of any application running on the authors' machines (not just browsing).

The above measurements from a single recursive (in both experiments) demonstrate that the impact of root DNS latency on users is small (a percent of page load time). However, the impact of root DNS latency on user performance is still not as small as one would expect. For example, there are up to 900 queries per day to the root server for the COM NS record at the research lab. Given the 2 day TTL of the COM record, 900 queries per day is unexpectedly frequent. We explore this issue further in Appendix C and find such problems may be caused by software bugs. This unexpectedly large number of queries suggests arguments that users rarely experience root latency because cached TLD records have long TTLs are not sufficient. To obtain more accurate estimates of what root DNS latency users experience, we next take a global view of root DNS querying behavior.

A Global Perspective of Root DNS Latency

Towards obtaining a global view of how users interact with the root DNS, we look at global querying behavior

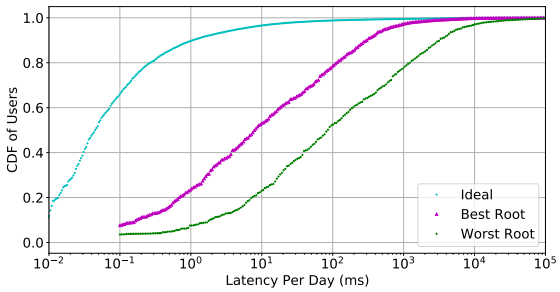


Figure 2: A CDF of approximate latency a user experiences due to root DNS resolution, per day, where the lines represent different ways of estimating root latency. Even using the worst (highest latency) root server, most users experience no more than 85 ms per day of root latency.

of recursives. As discussed in Section 2.3, 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 `ACDN` user data.

Given query volumes towards root servers from recursives and user counts behind each recursive from the DITL captures (§3.1), we estimate the amount of latency users experience per day due to root DNS resolution. Figure 2 is a CDF of expected user latency per day, where the expected value is calculated according to certain assumptions we make about root latency (we enumerate these assumptions below). We make these assumptions since it is difficult to capture global anycast latency to all roots. Figure 2 demonstrates that most users spend no more than 85 ms of latency per day waiting for root DNS queries, regardless of which method is used to estimate daily latency.

To generate each line in Figure 2, we multiply the expected root latency per query for each recursive by the number of queries per day each recursive makes and divide (*i.e.,* amortize) by the number of users that recursive represents. We estimate the expected root latency per query for each recursive in a few different ways which we describe below. This product (representing daily query latency 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, and we use user counts from a global CDN, Figure 2 provides

a truly global view of how users incur latency due to the root DNS.

Since we do not know the actual latencies from each recursive to each root, we estimate expected latency in multiple ways. First, we obtain latencies from RIPE Atlas probes to root servers during the same time as the DITL captures. The lines labelled "Best Root" and "Worst Root" in Figure 2 correspond to cases in which the latency from every recursive to every root are 15 ms and 159 ms, respectively. These latencies are the median (across RIPE probes) latencies to F and B root, and are the lowest and highest such medians measured using RIPE Atlas. Comparing Best Root and Worst Root demonstrates that the choice of root server makes little difference to the user (roughly 100 ms/*day* at the median), despite the "much better" performance of the Best Root. Put another way, the median user would only query the roots once per day (due to caching), and so would only experience the difference between the Best Root and Worst Root once per day.

The line labeled 'ideal' does not use DITL query volumes to calculate daily user latency, 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. For the ideal line, we assume latencies from all recursives to all sites are 15 ms (Best Root, for comparison). The resulting hypothetical median daily latency of 0.044 ms could represent a future in which caching works at recursives optimally – not querying the roots when not necessary. The 'ideal' line also demonstrates the degree to which the assumption that recursives should only query once per TTL *underestimates* the latency users experience due to the root DNS.

The comparison between the Worst Root and Best Root lines in Figure 2 reveals an important fact – reductions in root DNS latency *do not matter* for users. Specifically, a hypothetical ten-fold decrease in latency (*i.e.,* due to investment in new sites around the globe) provides minimal latency savings, on the order of tens of milliseconds per day, to users. Hence, not only does root DNS latency not affect users, but also the recent large increase in the number of root DNS sites has likely not been motivated by latency improvement.

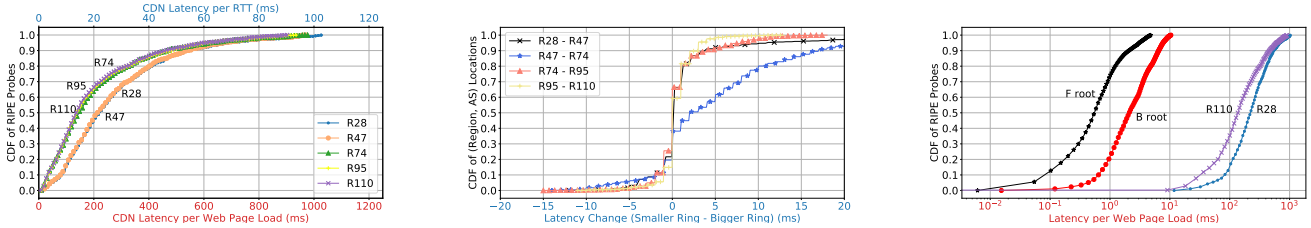
4.2 Impact of ACDN Latency on Users

We now measure how users are impacted by latency of ACDN, both to provide context for our results on inflation in Section 5 and to conduct the largest study to date of anycast CDN latency. Using both client-side measurements and server-side logs, we demonstrate that anycast latency results in orders of magnitude more delay to users for page loads from ACDN than users see due to the root DNS, and that latency usually decreases with more sites. Consequently, investments in more anycast sites positively affect user experience much more in the case of ACDN. This result is not surprising, but it is informative to explicitly compare performance differences between CDNs and the roots, and quantify how much more CDN latency matters to users than root DNS latency.

ACDN has *rings* that form a logical hierarchy of layers. Each larger ring adds some sites to those of the smaller ring. Rings use split TCP connections, providing front-ends to cloud services for content. Since each ring provides an IP anycast CDN, we report results for each of the rings individually. As a result, 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 not only what the RTT is from users to front-ends, but also how many RTTs are incurred when fetching a web object. Since it is difficult to measure the number of RTTs with broad coverage, we use a lower bound which suffices for our needs. The minimum number of RTTs per page load depends on the size of the page requested and the server's TCP implementation. We estimate RTTs per page based on the top-1000 pages according to GTMetrix [2] and web pages hosted by ACDN, taking an estimate of RTTs required to retrieve web page data. We find it usually takes at least 10 RTTs to load a web page, and so use this number in what follows. We describe these measurements and the calculation of RTTs from page sizes in further detail in Appendix A.4.

In Figure 3a, we show latencies to rings. Figure 3a and Figure 3c use latencies measured from RIPE Atlas probes, since we can not share absolute latencies from ACDN measurements. 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. We observed that the distribution of RIPE Atlas probes latencies is overall somewhat lower than that of ACDN's users globally (not shown in figure), so Figure 3a likely underestimates the latency users typically experience. Therefore, unsurprisingly, latency to ACDN factors into user experience. The difference in median latency per page load experienced by RIPE probes between R28 and



(a) (b) (c)

Figure 3: RTTs/latencies per web page load from RIPE probes to ACDN rings (3a), change in median latency for ACDN users in ⟨region, AS⟩ locations when transitioning rings (3b), and latencies per page load for B root, F root, R28 and R110 on a log scale (3c). Axes with per-RTT latencies are blue, while axes with per-page-load latencies are red. Performance generally improves as sites are added (3a, 3b), and ACDN incurred latency is orders of magnitude higher than root DNS latency (3c).

R110 is approximately 100 ms, which is a measure of how investments in more front-ends can help users.

Figure 3b shows change in latency for ⟨region, AS⟩ locations as ring size increases, calculated using ACDN measurements. 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. Figure 3b uses client-side measurements, so we do not know why latencies change when moving from ring to ring, because we do not know which front-ends clients use. However, server-side logs allow an educated guess. For example, R74 has a front-end in Atlanta, Georgia while R47 does not. Server-side logs show some users in San Antonio get directed to Atlanta when querying R74, missing the nearby front-ends in Texas.

4.3 Understanding Key Differences

To compare ACDN latency with root DNS latency, in Figure 3c we include per page load latencies for F root, B root, R28, and R110. We chose these two root letters as they provided the best and worst median latency for RIPE probes, respectively. We calculate page load latencies for ACDN as in Section 4.2. To calculate the number of root DNS requests in a page load, we multiply the root cache miss rate seen in our local experiments (Table 1) by an estimate of the number of serial DNS requests (S) in a page load. To measure S, we perform page loads for the 1,000 most popular domains, taken from GTmetrix [2], using the Selenium web-browser and calculate the number of blocking DNS requests per page load using previously developed methods [57]. 95% of page loads result in $S \leq 3$, and so the number of root DNS requests in a page load is 0.015 * 3 = 0.045. 0.045 is an overestimate, since 0.015 is an overestimate (§4.1) and we use a tail value of S.

Clearly ACDN latency matters *orders of magnitude* more than root DNS latency, no matter which percentile of which curve we look at in Figure 3c. We find it informative to explicitly show the difference in quality of experience implications, even though the result follows from the number of CDN RTTs and root queries per page load. The number of RTTs per page load is a key difference between these two services. Additionally, benefits of a larger root DNS deployment (F root vs B root) are only a few milliseconds per page load, while for ACDN a larger deployment (R28 vs R110) can save of hundreds of milliseconds per page load.

Figure 3 is a simple, effective way of demonstrating how users experience latency from ACDN, how much ACDN latency "matters", and how much more it matters to users than root DNS latency. This observation provides context for our investigation of inflation in these systems, in light of how overall system performance translates to user experience.

5 REASSESSING INFLATION

Earlier work has found queries to the root DNS are often significantly inflated [20, 21, 27, 28, 40, 51]. This section investigates whether root DNS inflation is a consequence of the fact that root DNS latency hardly matters (§4.1), or if substantial inflation is an unfortunate side-effect of using anycast to distribute content on the Internet. To answer this question, we directly compare inflation in the root DNS and in ACDN. We find that, while over 97% of queries to the root DNS are inflated on average, as few as 35% of users of ACDN experience any path inflation, and users experience much less inflation per round trip than in the root DNS. We then investigate routing in the root DNS and ACDN and provide evidence suggesting that extensive peering and engineering effort help ACDN to keep inflation low. Our study builds on prior work [20, 27, 28, 40, 51], presenting a global view of inflation across ten out of thirteen root DNS deployments. We

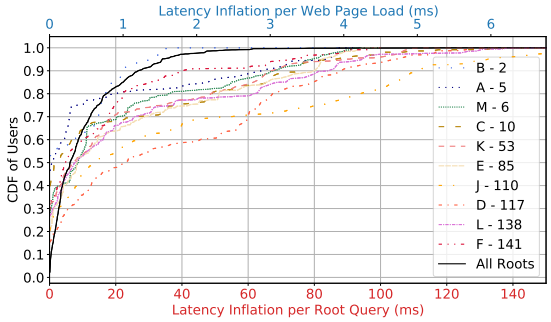


Figure 4: Inflation per root query (bottom axis) and per page load (top axis) for root deployments, and globally across all roots, experienced by users of ACDN. Generally, larger deployments are likely to inflate queries. The legend indicates the number of sites per letter during the 2018 DITL.

also present the largest study of CDN inflation to date, representing global measurements from more than a billion users in hundreds of countries and thousands of ASes.

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 [20, 21, 28, 40]. We choose to take a holistic approach to studying inflation, considering inflation relative to the overall deployment, rather than trying to infer which inflation would exist in an equivalent unicast deployment. This approach has two benefits. First, we are primarily interested in comparing performance of the root DNS versus CDNs. For ACDN, since overall inflation is small, both types of inflation are small. Second, overall inflation is easier to measure. 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 measure because some letters do not publish their unicast addresses, and measuring from many vantage points (*e.g.,* RIPE Atlas probes) to all sites would stress measurement budgets. Moreover, systems such as RIPE Atlas do not have representative coverage [18].

5.1 Inflation in the Root DNS

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

Figure 4 is a CDF of expected inflation per root query (bottom axis) and per page load (top axis) for users in the DITL∩ACDN data set for each letter individually, and globally (across all letters). To calculate inflation, we first geolocate all recursives in the DITL∩ACDN data set using MaxMind [4], matching prior methodology [40]. Major public DNS servers make server subnets and geographic locations publicly available [3], allowing us to (in addition) precisely geolocate 0.4% of (valid TLD) DITL query volume associated with these specific resolvers.

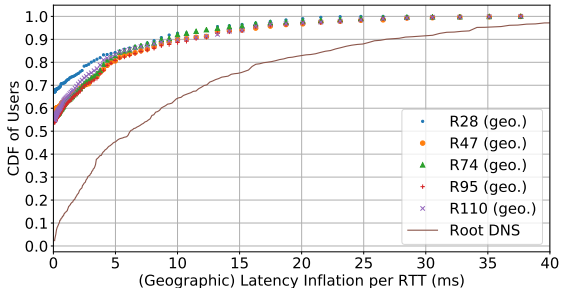
We then compute inflation for each recursive sending queries to root server j as

$$PI_j^g(R) = \frac{2}{c_f}\left(\sum_i \frac{n_q(R,j_i)d(R,j_i)}{N_j(R)} - \min_k d(R,j_k)\right) \quad (1)$$

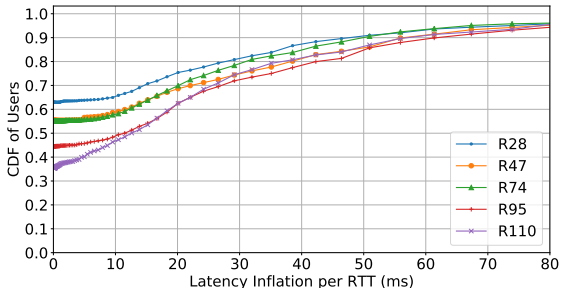
where $n_q(R,j_i)$ is the number of queries to site j_i by recursive R, $N_j(R) = \sum_i n_q(R,j_i)$ is the total number of queries to all sites j_i in root j by recursive R, c_f is the speed of light in fiber, the factor of 2 accounts for the round trip latency, $d(R,j_k)$ is the distance between the recursive resolver and site j_k, and both the summation and minimization are over the sites in this letter deployment. $PI_j(R)$ 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 latency inflation of a recursive is then the empirical mean over all roots. Although we look at geographic inflation, this simplification suffices since we are more interested in how root inflation compares to CDN inflation, how many users are being inflated, and how inflation compares among root deployments.

Figure 4 demonstrates that the likelihood of a root DNS query experiencing any inflation (y-axis intercept) roughly grows with deployment size, with a few exceptions. This finding corroborates results presented in prior work [40]. The 'All Roots' line captures the queries from each recursive to all roots. It has the lowest y-intercept of any line in Fig. 4, 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, routing to the roots is worse than previously thought – nearly every user is (on average) likely to experience inflation when querying the root DNS, and 15% of users are likely to be inflated by more than 2,000 km (20 ms).

As a useful visualization, we also display inflation *per page load* caused by root DNS queries on the top axis of Figure 4. Although the fact that perceived root DNS inflation is small is an obvious result, given our discussion



(a)



(b)

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.

in Section 4.1, we find it informative to explicitly place root DNS inflation in context, given the attention placed on this inefficiency in prior work [20, 21, 27, 28, 40, 51]. We use the same methodology to calculate page load metrics as in Figure 3c. 99.9% of users querying the root experience less than three milliseconds of inflation per page load, and the difference between the most inflation (J root) and the least inflation (B root) is only five milliseconds in the tail of per-page-load time.

5.2 Inflation in ACDN

We next investigate anycast performance in ACDN to quantify path inflation, and discuss its impact on user experience. Comparison to root anycast (§5.1) can help identify how design decisions differ by service, and how users of different services benefit from different choices.

To measure anycast inflation for ACDN we use geographic information, and server-side and client-side measurements (§3.2). Server-side logs are valuable for calculating path 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.

In Figure 5a and Figure 5b, we show two ways of measuring inflation for users of ACDN: geographic inflation and path 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⟩. ACDN user locations are determined using an internal database.

ACDN users usually experience no geographic latency 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. Although inflation in the roots does not matter for the user (§5.1), the fact that it is much larger and more prevalent than ACDN inflation per RTT (at every percentile) suggests ACDN optimizes its deployment to control it (§5.3).

We next calculate path inflation for each ring. ACDN measures actual latency from users to front-ends. 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.

Even though users from the same ⟨region, AS⟩ are usually routed together, they may occasionally be routed to different sites (thus inflating paths) due to load balancing in intermediate ASes, so we average inflation across sites for a ⟨region, AS⟩. For example, a load balancer may change the border router at which a packet egresses an edge network, thus potentially changing the ingress point into ACDN network. We calculate latency inflation $PI^{l}_{\texttt{ACDN}}(L, r)$ for users of ⟨region, AS⟩ L to ring r as

$$PI^{l}_{\texttt{ACDN}}(L, r) = \sum_{i} \frac{l_{r_i}(L) N_{r_i}(L)}{N_i(L)} - \frac{3 \times 2}{2 c_f} \min_{k} d(L, r_k) \tag{2}$$

where N_{r_i} is the number of users in L that visited front-end r_i, $N_i(L)$ is the total number of users in L who visited r, $l_{r_i}(L)$ is the median latency of users in L towards front-end r_i, c_f is the speed of light in fiber, and $d(L, r_k)$ is the distance from L to r_k. 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 [38], so we use this to approximately lower bound the best latency users could achieve. (We do not know optimal latency for users.)

As with root DNS, we see greater inflation as the number of front-ends grows (Fig. 5a). For example, in

R28, approximately 35% of users experience any inflation, while in R110, 65% of users experience some inflation. However, the CDN is able to keep *all* path inflation below 30 ms for 70% of users in all rings, lower than root DNS inflation reported in prior work [27, 28, 40]

5.3 Understanding Key Differences

Inflation is much lower for ACDN than in the roots, and ACDN users are inflated much less often than users querying the root DNS. Root DNS is critical infrastructure for which availability and security are paramount, and so we expect root deployments to optimize these properties. Inflation does not directly impact availability, and Section 4.1 demonstrated that root latency has little impact on user performance, and so operators need not eliminate inflation. 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 [53, 63], a global SD-WAN [33, 35], and expensive peering agreements – all of which are critical for ACDN's business but may not make sense for roots. Whereas many roots have an open peering policy, ACDN has a restrictive peering policy, only peering when it makes economic sense or helps 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 [44, 61]. With fewer BGP decision points, paths are often less inflated [54]. This intuition motivates the following investigation of AS path lengths towards roots and ACDN and of how path lengths relate to inflation.

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. Lengths towards ACDN are based on traceroutes from each active RIPE Atlas probe during August 2020, whereas lengths towards the roots are based on traceroutes from each RIPE Atlas probe during April 2018 (the time of DITL).[4] We then perform IP to AS mapping using Team Cymru [5], remove IPs that are private, associated with IXPs, or cannot be mapped to ASes. We merge AS siblings together into one 'organization'. We derive sibling data from CAIDA's AS to organization dataset [6]. We group paths by ⟨`region, AS`⟩, except for 'All Roots', for which we group paths by ⟨`region, AS, root`⟩.

[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.

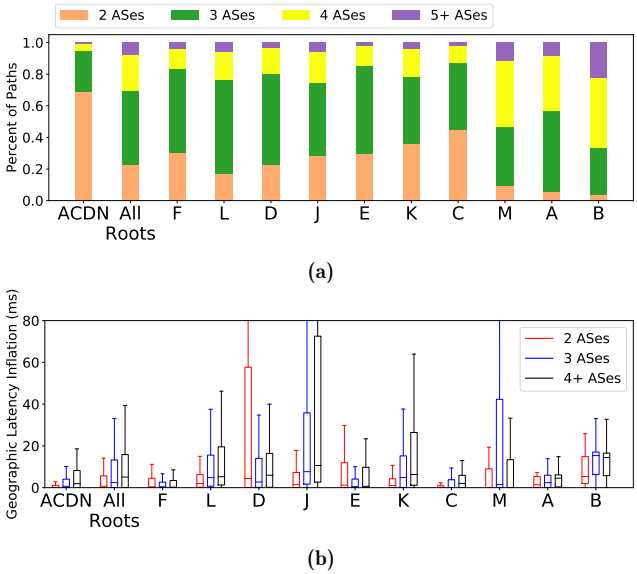


Figure 6: Distribution of the number of ASes traversed to reach various destinations (6a) and the correlation between the AS path length towards a destination and geographic latency inflation (6b). ACDN is closely connected to many eyeball ASes, and this connectivity correlates with lower inflation.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

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% of ⟨`region, AS`⟩ locations are directly connected to root letters 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. A ⟨`region, AS`⟩ pair's inflation towards a destination is the geographic latency inflation associated with that ⟨`region, AS`⟩ (Fig. 4, Fig. 5a), and 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 organizations 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.

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

ACDN puts toward peering and optimizing its routing and infrastructure. In addition to shorter AS paths generally being less inflated [54], ACDN's direct paths in particular sidestep the challenges of BGP by aligning the best performing paths with the BGP decision process [25]. 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 [54]. 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. Figure 6 quantifies one key difference between root DNS and CDN deployments, but publicly available data cannot capture all of ACDN's optimizations.

6 RELATED WORK

Root DNS Anycast. Prior work looks at anycast's ability to load balance and defend against DDoS attacks [47, 51]; we do not consider anycast's performance in this context. Many prior studies look at latency and inflation performance in the root DNS [28, 40, 41, 51, 51], however none relate performance to user experience, nor do they compare performance to other systems.

CDN Anycast. Some CDNs use IP anycast [17, 21, 59]. Some prior work looked at inflation in CDNs [21], finding it to be similarly low. Our work presents a much larger study of latency and inflation, places performance metrics in the context of user experience, and compares performance to other systems that use anycast.

Recursive Resolvers and the Benefits of Caching. Prior work has looked at statistics and latency implications of local resolvers [23, 36]. 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 [31, 39, 58, 64]; 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 [10, 19, 57], 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 [16].

7 DISCUSSION AND CONCLUSION

Anycast performance is interesting to assess in its own right. However, the magnitude of *problems* caused by anycast's inefficiencies are proportional to their effect on end users. Conclusions we draw about "what should be done" in the face of such inefficiencies depend on the system in which anycast is used. This principle extends beyond anycast and should be applied to any technique.

Recent work suggested that inflation is a serious problem, based on analysis of root DNS [40]. The paper argued that adding more sites may hurt rather than help deployments (due to inflation), and suggested that solutions either involve cooperation from a large ISP or widespread modifications to BGP policy. The root DNS is a vital, heterogeneous anycast deployment, and data is easy to access. However, conclusions based on this data may not apply in other contexts beyond the root DNS.

We build on this work by investigating both root DNS and anycast CDNs. Considering the behavior of each deployment in the context of its role reveals a richer picture of anycast behavior than can be learned from studying root DNS alone, especially since inflation and efficiency depend mostly on deployment details.

Our analysis of ACDN data provides evidence that network operators can control anycast's inefficiencies, even though these inefficiencies are quite prevalent in the root DNS. Inefficiency does not tell a complete story, since the root DNS does not have a strong incentive to decrease latency. DNS caching means that users experience nearly no latency from root DNS queries, and even eliminating path inflation in the root is unlikely to be noticed. Conversely, the low latency and inflation achieved by ACDN results from extensive resources for infrastructure and peering, and the constant engineering, monitoring, and automation for network optimization and debugging. We hope others will take our results into consideration in future studies discussing anycast performance as a technique used by DNS and other systems.

REFERENCES

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

[2] 2019. The Top 1,000 Sites on the Internet. gtmetrix.com/top1000.html

[3] 2020. developers.google.com/speed/public-dns

[4] 2020. maxmind.com/en/geoip2-databases

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

[6] 2020. caida.org/data/as-organizations/

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

[8] 2020. Data center Locations. opendns.com/data-center-locations/

[9] 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

[10] 2020. The HTTP Archive Project. httparchive.org/

[11] 2020. IANA IPv4 Special-Purpose Address Registry. iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

[12] 2020. Root Servers. root-servers.org

[13] 2020. Tshark. wireshark.org/docs/man-pages/tshark.html

[14] 2020. Window: load event. developer.mozilla.org/en-US/docs/Web/API/Window/load_event

[15] 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

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

[17] Amazon. 2020. Amazon CloudFront. aws.amazon.com/cloudfront/

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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

[25] 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.

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

[27] 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).

[28] 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.

[29] 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.

[30] 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.

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

[32] 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

[33] 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.

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

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet anycast: performance, problems, & potential..

In *SIGCOMM*. 59–73.

[41] 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.

[42] Greg Linden. 2006. Make Data Useful. http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt.

[43] 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.

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

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

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

[47] 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.

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

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

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

[51] 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.

[52] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. 2019. Internet Performance from Facebook's Edge. In *Proceedings of the Internet Measurement Conference*. 179–194.

[53] 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.

[54] 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.

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

[56] Stoyan Stefanov. 2008. YSlow 2.0. In *CSDN SD2C*.

[57] 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.

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

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

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

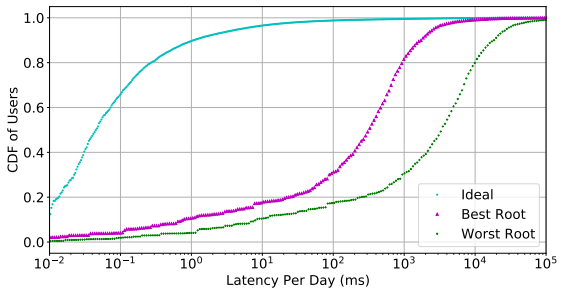


Figure 7: Daily latency experienced by users due to the root DNS, calculated by amortizing root DNS requests over user populations, when including or excluding queries for invalid TLDs. Counting invalid queries drastically increases median daily user latency estimates to 327 ms for Best Root (a 30-fold increase).

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 pre-processing steps, analyze the effects of these assumptions on our results, and, when possible, offer alternative methods of performing the analysis to demonstrate that results remain consistent.

A.1 Effect of Removing Invalid TLD Queries

In Section 4.1 we estimate the amount of latency 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 latencies (*i.e.,* towards a recursive's daily query count), 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 median latencies due to root DNS resolution by 30×.

We base this decision on prior work on 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 [15]. Researchers at USC have found that more than 90% of single-label queries at the root match the Chromium captive-portal pattern [31]. 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 experience latency and 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, others have found that approximately 60% of queries for invalid TLDs reaching root servers are for domains such as local, no_dot, belkin, and corp [29]. 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 the 'Best Root' latency, users experience 327 ms of root latency at the median – about 30× more than when we exclude requests for invalid queries (§4.1). This drastic 30-fold difference 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. For example, there is a 3 *second* difference at the median between Best Root and Worst Root. This 3 second difference is harder to dismiss than the corresponding 78 ms difference between Best Root and Worst Root medians in Figure 2. However, this three seconds *per day* is still quite small (less time than a single page load [10]), so the major conclusions reached in Section 4.1 still apply.

A.2 Representativeness of Daily Root Latency Analysis

In Section 4.1 we estimate the amount of latency 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 (§3.1). Naturally recursives used by users of

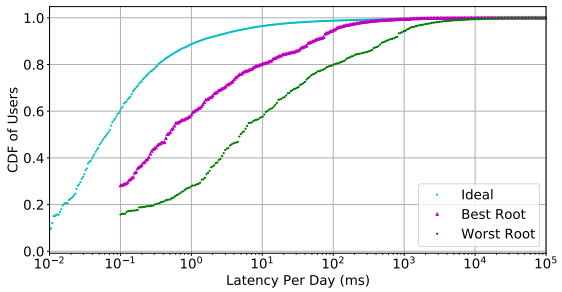


Figure 8: A CDF of approximate latency a user experiences due to root DNS resolution, per day, without joining recursives in DITL with recursives seen by ACDN by /24. This unrepresentative analysis yields an estimate of daily user latency far lower than in Section 4.1.

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 2: 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%.

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 now justify this decision and discuss the implications of this preprocessing step on the results presented in Section 4.1.

In Table 2 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 2, in Figure 8 we show daily latency experienced by users of ACDN due to root DNS latency *without* aggregating query and user statistics by /24. The median daily latency when users use 'Best Root' is only .46 ms – 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 2).

Table 2 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 daily latency experienced by users due to 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 [60], 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

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

where j_F is the favorite site for /24 k (*i.e.,* the site the /24 queries the most), and Q^k is the total number of queries from /24 k. We plot these fractions for all /24s in DITL, and for each root deployment.[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 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 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

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

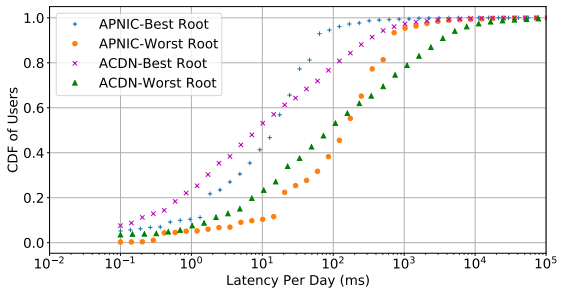


Figure 10: Daily latency experienced by users due to the root DNS, calculated by amortizing root DNS requests over user populations. The APNIC and ACDN lines use two different methods of estimating user counts behind each, but arrive at the same basic conclusion – users do not experience much latency due to the root DNS.

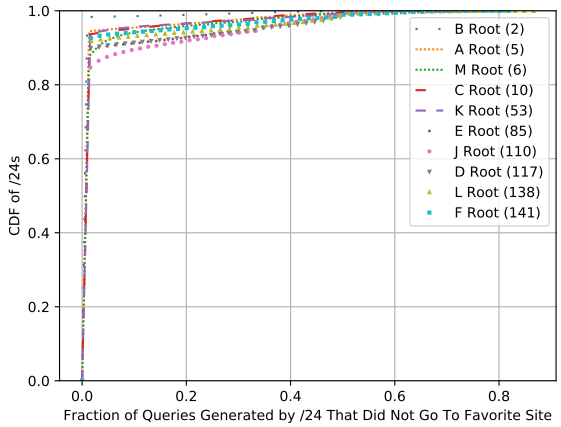


Figure 9: Fractions of queries generated by /24s that do not hit the most popular site for each /24 and for each root letter in question. For all root letters, more than 80% of /24s have all queries visit the most popular site, suggesting queries from the same /24 are usually routed similarly.

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).

Appendix A.3 offers further evidence that this decision is justified, since it corroborates our results regarding daily user root latency using very different methodology. Specifically, without relying on aggregating statistics by /24 or using ACDN user counts, we reach similar quantitative and qualitative conclusions about daily user latency experienced due to the root DNS.

A.3 Revisiting Root Latency Experienced per Day

In Section 4.1 we estimate the amount of latency users experience due to the root DNS by amortizing queries over user populations. Unfortunately, we could not obtain user population estimates for every recursive resolver, so we focused on a subset of resolver for which we did have population data.

To complement the analysis in Section 4.1, we use Internet user population estimates by AS from APNIC [34] instead of ACDN user counts to amortize root DNS latency . These 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 when generating the APNIC user counts are that Google Ad placement is uniform across ASes within a country, and that both an AS and its users are fully contained in a single country.

We use this dataset to amortize root DNS queries seen from ASes over user populations. First, we use the TeamCymru IP to ASN mapping to map IP addresses seen in the DITL captures to their respective ASs [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. Next, assuming recursives only serve users in the same AS, we divide the number of daily queries seen in DITL from each AS by the number of users in that AS to obtain the number of daily queries per user. 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.

We plot the resulting latencies in Figure 10 for Worst Root and Best Root, and include the corresponding lines using ACDN user estimates for reference (§4.1).

Using APNIC's user estimates, users experience approximately the same daily user latency at the median (about 15 ms for Best Root), and much less daily user latency in the tail. This result is interesting since the same conclusions can be drawn about daily root DNS latency experienced by users using two very different methods of amortizing root latency over user populations. This method of computing root DNS latency experienced by users reaffirms that users do not experience much latency due to the root DNS.

A.4 Estimating the Number of RTTs in a Page Load

To estimate the latency a user experiences due to anycast (§4.2), we first must estimate the number of RTTs required to load a typical web page hosted by ACDN. We provide an estimate of the number of RTTs based on

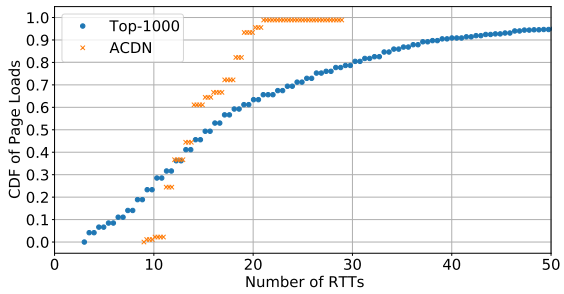


Figure 12: The number of RTTs for page loads. ACDN and Top-1000 refer to the types of pages being loaded. Only a few percent of ACDN web pages are loaded within 10 RTTs and 90% of all page loads are loaded within 20 RTTs.

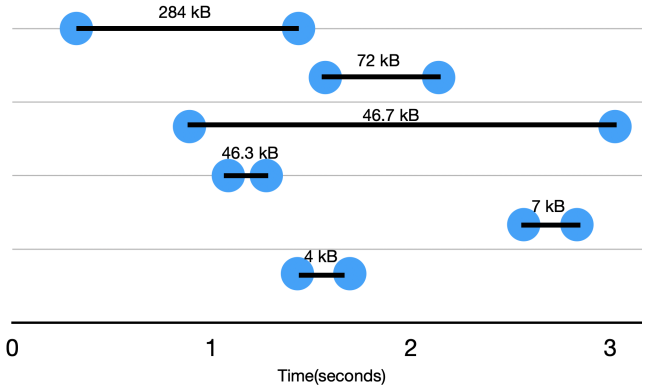


Figure 11: Example TCP connections for a web page loading. Each line is a TCP connection. Disregarding smaller downloads that overlap longer ones, we choose to accumulate RTTs associated with the 284 kB, 72 kB and 7 kB downloads.

modeling and evaluation of a set of 1,000 web pages using Selenium (a headless web browser).

ACDN latency occurs when a user downloads web objects via HTTP. 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

$$N = \lceil log_2 \frac{D}{W} \rceil \quad (4)$$

where N is the number of RTTs, D is the total number of bytes sent by the TCP connection from the server to the user, and W is the initial congestion window size in bytes [24, 32]. Although W is set by the server, ACDN and a majority of web pages [50], set this value to approximately 15 kB so we use this value. (We do not consider QUIC or persistent connections in detail here, but larger initial windows will result in fewer RTTs.)

In Section 4.3 we aim to show ACDN latency per page load is orders of magnitude more than root DNS latency per page load, so we make the following assumptions to establish a *lower bound* on N:

(1) The connection is limited by W, as opposed to the user's receive window. Connections are often limited by processing resources and serial requests [52], rather than the congestion window.
(2) TCP is always in slow start mode, which implies the window size doubles each RTT.
(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 often 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 by the user. 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. This process of 'pruning' connections is illustrated in Figure 11. Disregarding the smaller download sizes, we choose to accumulate RTTs associated with the 284 kB, 72 kB and 7 kB downloads.

We load two 'classes' of web pages: the top-1000 web pages according to GTMetrix [2] (once for each page) and nine web pages owned by ACDN that we study in this paper (twenty times for each page). We use Selenium and Chrome to open web pages and use Tshark [13] to capture TCP packets during the page load. For each web page, we only capture packets until the loadEventEnd event has been fired by the browser. When the load event ends, the whole page has loaded, including all dependent resources such as stylesheets and images [14]. To calculate the total data size for each connection, we use the ACK value in the last packet sent to the server minus the SEQ value in the first packet received from the server. We set W to 15 kB since a large fraction of web pages use this value in practice [50]. We then calculate the number of RTTs using Equation (4), and add a final two RTTs for TCP and TLS handshakes.

The RTT estimates are shown in Figure 12. As Figure 12 demonstrates, only a few percent of ACDN web pages are loaded within 10 RTTs, and 90% of all page loads are loaded within 20 RTTs. Since we try to obtain a lower bound on the number of RTTs per page load, we believe we are justified in using 10 RTTs in our analysis in Section 4.2.

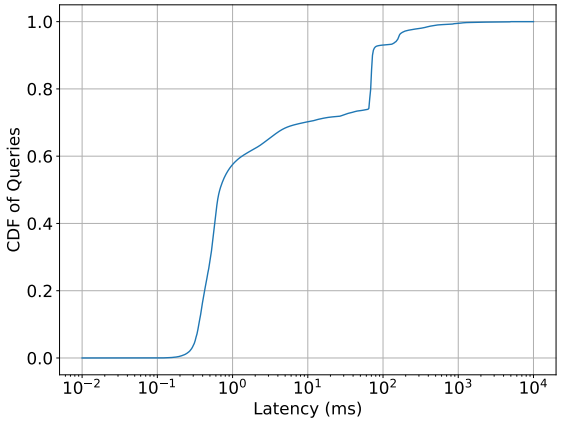


Figure 13: CDF of user DNS query latencies seen at a recursive resolve at ISI, over the course of one year. Latencies are measured from the timestamp when the recursive resolver receives a client query to the timestamp when the recursive sends a response to that client query. The sub-millisecond latency for more than half of queries suggests most queries to this recursive are served by the local cache.

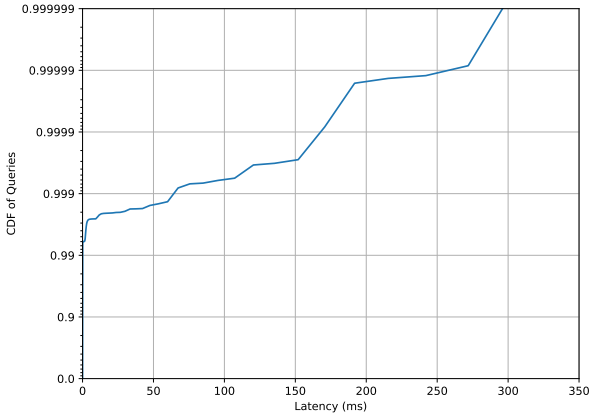


Figure 14: Root DNS latency for queries made by users of the research lab recursive resolver during 2018. This plot demonstrates the benefits of crowd-sourced caching and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than .1% incur latencies greater than 100 ms. User queries that did not generate a query to a root server were given a latency of 0.

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 13 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 [23]. 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 4.1, root DNS queries make up a small fraction of all queries shown in Figure 13. To visualize just how small this fraction is, Figure 14 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 14 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.

C CASE STUDY: REDUNDANT ROOT DNS QUERIES

When we investigate the traffic from a recursive resolver to the root servers in Section 4.1, 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 [29].

To observe these redundant requests in a controlled environment, we deploy a BIND instance (the resolver

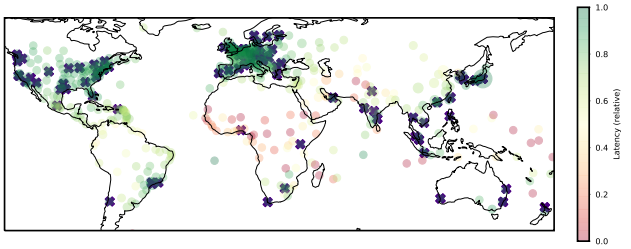


Figure 15: A visualization of front-ends in R110 (purple Xs), and user populations (transparent circles). User populations are colored according to the relative latency they see, and have size proportional to user population. Red corresponds to high latency and green corresponds to low latency. Latency generally gets lower, the closer users are to a front-end, and front-ends are concentrated around large user populations.

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 3: Pattern example of the redundant root DNS requests. The last 5 requests to J root are redundant which may be caused by an unanswered request in step 4.

in Section 3.1 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 [2] using Selenium and headless Chrome, While loading web pages, we collect network packets on port 53 using Tshark [13].

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 3.

In Table 3, 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 3, 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 (§4.1), 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 2, 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 15 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, 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.