A Tale of Two Anycasts

Latency and Efficiency of Root DNS versus CDNs

2020-07-29 draft, not for redistribution

Thomas Koch
Columbia University

Ke Li
Columbia University

Ethan Katz-Bassett
Columbia University

Matt Calder
Microsoft

John Heidemann
ISI

ABSTRACT

Anycast is widely used today to provide many types of content, including web-content from CDNs or services such as DNS. Recent years have seen huge investments in anycast infrastructure. In April 2020, root DNS is provided across more than 1000 sites, multiple CDNs have more than 1000 sites, and "small" CDNs often deploy several dozen sites. Yet research has suggested that anycast path inflation is a problem with users often routed to suboptimal sites, often based on analysis of root DNS. Is this huge investment in many anycast sites worthwhile for these applications? Is path inflation visible to users? Do studies of path inflation for root DNS apply to other services? We answer these questions with experiments using global traces from a large anycast CDN and root DNS traffic. We show that, while latency is important to CDNs, it is largely irrelevant for root DNS – even with imperfect anycast routing, most users see less than 15 ms per day from root DNS because caching is so effective. Second, while path inflation suggests that relative latency is not always optimal, we show that absolute latency overall is quite good. These trends increase as the number of sites grow: more users see imperfect routes, but absolute latency improves. While anycast path inflation happens, it is not a problem that hinders users of the DNS root today, and it does not severely degrade CDN performance.

1 INTRODUCTION

IP anycast, an approach to routing in which geographically diverse servers known as anycast sites all use the same IP address, is used by a number of operational Domain Name System (DNS) [3, 6, 8, 11, 49] and Content Delivery Networks (CDNs) [15, 22, 55] systems today, in part because of its ability to improve latency to clients and decrease load on each anycast server [37, 44, 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 [22, 40, 43, 51, 57]. Results have suggested that routing is potentially inefficient, sometimes not reaching the physically nearest anycast instance and thereby providing higher latency than would be optimal [40, 41, 51]. However, the degree to which this potential 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 and end-user performance—how does anycast architecture and efficiency affect the end-user experience? To answer this question, we analyze anycast with two different, real-world applications: the root DNS and a large anycast CDN, chosen for their overlapping, yet distinctive, goals. Data from the root DNS (e.g. packet traces) are readily available [11], and feature in existing anycast studies[28, 29, 40, 43, 46]. In addition to availability of data, the 13 root services operate independently with diverse deployment strategies.

We also examine a major commercial CDN. Deployed as multiple rings of different sizes, the CDN has some deployment diversity, although all operated by one organization. CDNs typically serve HTTP traffic, so the traffic size is larger and interaction with users more frequent than in the root DNS. These differences heavily influence the conclusions we can draw – while we find that mitigating anycast path inflation is quite important for anycast CDNs, the impact of latency in the root DNS setting is negligible.

To understand how anycast affects performance in these applications, we first look at typical latency for these services. We find that root DNS latency can be quite low, but varies across different root services. However, these differences are hardly perceived by users–we show they account for a few milliseconds of wait time per day, or fractions of a percent of a page load (§4.1). Delay is minimal due to caching of root DNS records with long TTLs at recursive resolvers. By contrast, greater latency to users using websites backed by the CDN can result in orders of magnitude more latency over the day than in the root DNS (§4.2). Hence larger deployment sizes can provide tangible latency benefits to anycast CDNs, but provide little latency benefit in the root DNS setting, and the magnitude of these benefits depends on deployment details. Collectively, these results suggest organizations managing anycast services have different incentives and strategies when deploying networks, which should be taken into account when analyzing anycast inflation and efficiency.

With these latency results, we revisit frequently posed questions about how anycast impacts services. We investigate anycast path inflation prevalence in deployments, and how path inflation affects users browsing the web. For root servers, we find that increasing deployment size can lead to more users experiencing path inflation; however, the *amount* of inflation is tiny (§5.1). Conversely, we find that, for an anycast CDN, the latency per page load can decrease by hundreds of milliseconds when adding more sites, even though this increase results in more path inflation (§5.2). Moreover, regardless of deployment size, the path inflation per RTT for the anycast CDN is about half that of the root deployments. Hence, even with greater path inflation, the latency benefits of more sites help CDNs and do not appreciably hurt DNS, suggesting that minimizing path inflation should not be a primary goal in anycast operation.

Recent work has suggested that anycast inefficiency (path inflation) is a serious problem, based on analysis of root DNS deployments [40]. We build on this work by investigating two classes of anycast applications: root DNS and Content Delivery Networks. 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. Our paper reveals that *anycast is what you make of it.* The root DNS is a common subject of anycast studies because it has readily available data, but results from the root DNS should be interpreted in the context of its role. Systems such as the root DNS do not have a strong incentive to decrease latency, since 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, latency is critical for CDNs where data is often tens or hundreds of KB in size. We find both low inflation and latency in an anycast CDN, because the CDN engineers its network to reduce latency and inflation. The fact that the impact of anycast performance is application-specific should factor into suggested improvements for anycast, anycast deployments, and Internet policies.

2 BACKGROUND: ANYCAST IN DISTRIBUTED SYSTEMS

IP anycast is a system in which geographically diverse servers known as anycast sites provide the same service at one common IP address. With the same IP address announced from many locations, the Border Gateway Protocol selects its best route from each user to one of the sites. Next, we review the pros and cons of anycast for DNS and CDNs.

2.1 Potential Benefits of Anycast

IP anycast is simple to deploy and scalable to the number of sites. Network managers offload the responsibility of mapping users to sites to the network. It thereby simplifies site maintenance, allowing addition of new sites or withdrawal for maintenance to happen automatically with BGP route changes. Anycast also provides resiliency in the face of unexpected site or network failures – should one site go offline, BGP will recompute routes and automatically point users to the next closest available site [44].

Anycast also helps latency and network capacity. Although BGP path selection is not designed to guarantee minimal latency, it does minimize BGP hop counts, a metric that is correlated with latency [29]. Anycast improves capacity by naturally distributing load across many locations [46].

2.2 Potential Drawbacks of Anycast

The two drawbacks of anycast are that it does not directly measure and optimize for any performance metric, and that it provides only limited control of load balancing.

Since packets are routed to destinations based on BGP's notion of best path (number of AS-hops), users may communicate with anycast sites that have a higher latency than the best alternative. This notion, anycast path inflation, is a concept introduced in prior work [29] and expanded upon in more recent work [40], which measures how inefficiently queries to anycasted IP's are mapped to physical sites. We use the definition from Li et. al. [40], which breaks down path inflation into unicast path inflation and anycast path inflation. Unicast path inflation measures the extra latency a query incurs as a result of not being mapped to the geographically closest physical site. Anycast path inflation measures the additional latency incurred beyond the lowest latency unicast alternative. Anycast path inflation can be both difficult to measure, since latencies to all unicast alternatives must be known and large, which (generally) occurs when queries are routed to geographically distant sites, despite the existence of a close site.

2.3 Root DNS anycast

DNS is a fundamental lookup service for the Internet, typically mapping hostnames to IP addresses [26, 45]. To resolve a name to its result, a user sends DNS requests to one or more recursive resolvers (recursives), usually provided by their ISP. The recursive then requests the records from a root DNS server, top level domain (TLD) server and authoritative DNS server corresponding to the record the user requested.

The root DNS servers are grouped into thirteen letters [11], each with its own anycast service on distinct IPv4 and

IPv6 addresses. They are managed by twelve distinct organizations. As of May 2020, each letter has anywhere from 6 to 252 anycast sites. The root DNS has more than 1000 top-level domain names (each with nameservers and glue records). Almost all have a cache lifetime of 2 days (2 records last 1 day). Hence, in an ideal world, every caching recursive would only need to query for each record once every one or two days – amortizing these requests over large user populations would imply users rarely incur root latency. We explore DNS request amortization in Section 4.1. When a recursive needs to query a root server, it may query whichever one it wishes (subject to network administration policy); however, recursives prefer lower-latency authoritative servers [47, 58]. The performance of anycast in the root DNS has been studied in a variety of contexts [28, 29, 40, 43, 46].

2.4 CDN Anycast

A second important application of anycast is content delivery. We study a large anycast CDN that serves web content to millions of users from more than 100 front ends. Traffic destined for the CDN enters its network at a point of presence (PoP), and is routed to one of the anycast sites serving the content (front-ends). The CDN has a logical hierarchy of layers, called rings, that conform to varying degrees of legal restrictions. Hence, traffic from a user prefix destined for the CDN may end up at two different front ends (depending on the service and location of the user), but often will ingress into the network at the same PoP. We use rings in this paper to simulate anycast deployments of different sizes, since each ring is a different size.

In Figure 1 we show the front-ends and user concentrations of the anycast CDN. Rings are named according to the number of front-end they contain, and Front-ends are labelled according to the smallest ring to which they belong (or else all front-ends would be labelled as R110). The CDN operator provides average user locations which we show as circles, where the radius of the circle is proportional to the population of users in that region. We can see that the deployment strategy of rings is to minimize latency to as many users as possible. We provide another visualization illustrating latency differences by region in Appendix D.

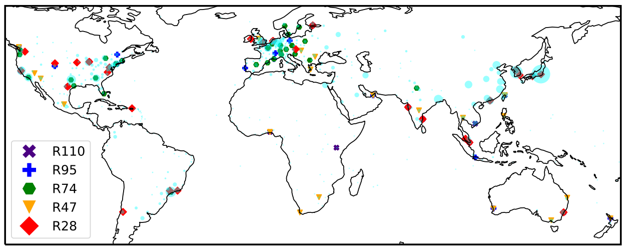


Figure 1: The hierarchical ring structure, and its global presence. User populations are shown as circles, with the radius of the circle proportional to the number of users in that region. The CDN has deployed front-ends in areas of high user concentration to provide low-latency options to as many users as possible.

3 METHODOLOGY AND DATASETS

We have one dataset for each of the services using anycast we study. The data we use to analyze the root DNS is readily available [1], while the CDN data is proprietary, gathered from an internal measurement system the CDN operates . We supplement these datasets with measurements from RIPE Atlas [52], a global measurement framework. As of May 2020, RIPE Atlas has more than 11k active measurement probes in hundreds of countries.

3.1 Root DNS Data and Methodology

To measure how root DNS latency impacts users, we would ideally track root DNS queries from the moment they are triggered (e.g., during a web page load), and determine when those DNS queries are in the 'critical path' of the process users are executing. However, with diverse operating systems and applications, many trying to reduce user-perceived latency, such a measurement is impossible, especially at scale. Instead, we rely on two DNS data sets: long-term packet captures from a research lab in a university in the United States, and from Day In the Life of the Internet (DITL), 48-hour packet captures at most root servers [1].

To obtain a local view of root DNS queries, we use packet captures from a recursive resolver serving users at a research lab in a university in the United States. 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 are relatively recent and overlap temporally with other datasets we use. This recursive resolver received queries from approximately 200 unique IP addresses each day over 2018. This recursive resolver serves some general 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 did examine the dataset for anomalies like measurement experiments and found none 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 Day in the Life of the Internet (DITL) captures, archived by DNS-OARC [1] (the most recent available when we began this work). DITL occurs approximately 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). IP addresses from B- and I-Root are partially or fully anonymized. Since we only comment on qualitative results (orders of magnitude), the exclusion of I Root and G Root

Data Set	**Statistic**	**Percent Overlap**
DITL ∩ CDN	DITL Recursives	29.3% of DITL Recursives
	DITL Volume	72.23% of DITL Query Volume
	CDN Recursives	78.8% of CDN Recursives
	CDN Volume	88.1% of CDN Query Volume
DITL ∩ CDN ∩ RIPE	DITL Recursives	.14% of DITL Recursives
	DITL Volume	20.7% of DITL Query Volume
	CDN Recursives	.34% of CDN Recursives
	CDN Volume	54.6% of CDN Query Volume

Table 1: Statistics displaying the extent to which the recursives of users in a large CDN overlap recursives seen in the 2018 DITL captures. Also shown is the extent to which recursives of RIPE probes represent the 2018 DITL captures. For example, the percent overlap of DITL recursives in DITL ∩ CDN is the number of DITL recursives in DITL ∩ CDN divided by the number of recursives in DITL.

traffic from analysis in Section 4.1 likely does not affect our high-level conclusions. All roots except H-root group the captures by site, giving us a global yet detailed description of what traffic is reaching various servers throughout the world.

We pre-process this dataset to remove queries we do not believe affect user latency. Of the 51.9 billion daily queries, 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 Google's Chromium browser [32], and so involve machine startup and not browsing latency. We believe most of the remainder are generated by other malfunctioning, automated software. Similarly, while PTR queries have some uses (traceroutes and confirming hostnames during authentication), they are not part of typical user web latency. We explore this decision, its effects on our conclusions, and provide further justification for this pre-processing step in Appendix A.1.

Sources of DNS queries in the DITL captures are, almost always, recursives, 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 recursives, gathered in 2019. This user data is from DNS data at a large CDN, and it counts unique IP addresses as "users". We recognize this definition may undercount multiple human users that use a single IP address with Network Address Translation (NAT). The large CDN maps recursives to user IP addresses by instrumenting users to request DNS records for domains the CDN controls when users interact with the CDN, a technique described in the literature [23, 42].

We then join the DITL captures and CDN user counts by the recursive resolver /24, aggregating the DITL query volumes from each /24 prefix and the CDN user IP counts seen using each /24 prefix.[1] We aggregate by /24 to increase the amount of recursives for which we have user data, noting that organizations may use many colocated servers using the same /24 as recursives [3, 7]. Prior work has also found that up to 80% of /24's are collocated [31]. We further explore this decision, explore its implications on our results, and provide further justification for this pre-processing step in Appendix A.2. For clarity, we henceforth refer to these /24's as recursives, even though each /24 may represent several recursives. We call this joined data set of query volumes and user counts by recursive DITL∩CDN. We next remove queries from prefixes in private IP space [10] (7% of all queries). Finally, we analyze only IPv4 data and discard IPv6 traffic (12% of queries) since we do not have IPv6 user data.

There is a mismatch in the recursives present in each data set. Table 1 summarizes the extent to which the CDN data (which is a subset of all users in the world) overlaps the DITL captures, and vice versa. Although the DITL∩CDN only represents 29.3% of all recursives seen in DITL, it captures a disproportionately large amount of all DITL volume (72.2%). However, we acknowledge that the lack of overlap summarized in Table 1 suggests the analysis we perform may not be representative. 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, lending credence to our overall conclusions about root latency experienced by users.

3.2 Anycast CDN Data Sources

To study performance in the anycast CDN, 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 to front-ends,[2] by user AS, location, and serving front-end. The CDN determines the location and AS of users using internal databases.

User locations are tracked by *region*, a geographic area used internally by the CDN 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 often 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:

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

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

130 in Europe, 60 in Africa, 100 in Asia, 1 in Antarctica, 130 in North America, 40 in South America, and 25 in Oceania.

The client-side measurements, described briefly in Section 3.1, come from a measurement system operated by the CDN, similar to those described in the literature [23, 42]. From this CDN measurement system, we collect latencies of users populations, noting the location and AS of the user. With this measurement system, we are able to hold the user population constant, for a given ⟨`region, AS`⟩, across rings. Holding user populations constant 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 millions of users across approximately 15,000 ⟨`region, AS`⟩ pairs.

We also use RIPE Atlas to ping rings, so we can directly compare latency to those collected by the CDN, to provide approximate latency (latency from the CDN is considered proprietary). In total, we collect 7,000 measurements from 1,000 RIPE probes in more than 500 ASes to augment the CDN latency measurements.

4 LATENCY IMPACT ON END USERS

To understand anycast performance, we first investigate latency for both the root DNS and CDNs. We quantify the latency users experience, how latencies between services compare to each other, and the effects additional sites have on user latency.

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 millions of users). Local evaluation estimates the fraction of page load time users wait for root DNS resolution, while global analysis estimates the total time per day users worldwide wait for root DNS resolution. We find that users likely spend less than 10 ms waiting for root DNS resolution during a page load (local view), and 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.

A Local Perspective of Root DNS Latency

We first quantify how much latency users see from root DNS latency during a web page load. We use packet captures of the recursive resolver at a research lab (§3.1). We do not claim the behavior at this resolver is globally typical of users, but believe we may still draw valuable qualitative results. We first calculate the number of queries to the 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. We say approximately since, for example, the recursive resolver may have sent multiple root requests per user query, or root requests without any user query triggering them.

Statistics	Number of User Queries (millions)	14.9
	Number of Root Transactions	73,200
Assumptions	Web Page Load Time (ms)	3,000
	Root DNS Latency (ms)	500
	Number of DNS Look-Ups Per Web Page	3
Implications	Percent of user Queries Resulting in a Root Transaction	0.5
	Expected Speed-up in PLT with No Root Latency (ms)	8 ms
	Resulting PLT Speedup (percent)	0.25%

Table 2: Root querying statistics gathered from the research lab recursive for a representative month of 2018, and associated implications of how root latency impacts users of ISI.

Table 2 gives relevant statistics and their implications on user-perceived latency. 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%, so we choose to include this median/global value in Table 2.

We use this root cache miss rate to approximate latency experienced by users during a web page load. Assume that a web page load takes W ms, that there are S serial DNS requests per page, that the root latency is l_r ms, and that the root cache miss rate is m. Then, the resulting (average) latency due to root DNS resolution per page load is approximately given by ml_rS,[3] which, as a fraction of page load time is $\frac{ml_rS}{W}$.

Although W and l_r depend on a number of factors (e.g. browser, web page size), m can be measured for each recursive resolver (as shown in Table 2), and S is usually small. We use $S = 3$ as a typical value. 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 [53]. We find 95% of page loads result in $S \leq 3$. As exaggerated upper bounds on the impact of root DNS latency, we choose $W = 3{,}000$ ms (a fast page load) and $l_r = 500$ ms (a slow root latency).[4]

[3]We are modeling cache hits as Bernoulli trials with parameter m. Since m is small, the probability of two or more cache misses in S DNS requests is negligible.

[4]According to HTTP Archive [9], RIPE Atlas [52], and the results from the research lab, these are quite conservative estimates in the sense that they *overestimate* root DNS latency's effect on PLT.

The upper bound on how much latency a user could save, if root queries were free, is only 8ms $0.005 \times 500 \times 3$). As a percentage of the total page load time, this latency is $\frac{8}{3{,}000} = 0.25\%$, which is measurable, but likely makes little difference to users. For comparison, a conservative estimate of the root cache miss rate of 2.5% (as opposed to .5%), would result in root DNS resolution comprising approximately 1.25% of a single page load (Table 2). In Appendix B we provide another visualization of the minimal impact root DNS latency has on users of the research lab, and all DNS latency users at the research lab experience.

The above demonstrates the impact of root DNS latency on users is small (a few milliseconds). However, the impact of root DNS latency on user performance is still not as small as one would expect. For example, a particular day during January 2018 sees as many as 900 queries to the root server for the COM NS record. Given the 2 day TTL of this record, 900 queries per day is an unexpectedly large frequency. 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 (as opposed to what they could *ideally* 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 of recursives. Given query volumes towards root servers from recursives and user counts behind each recursive from the DITL captures (§3.1) , we are able to 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). Figure 2 demonstrates that 50% of 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 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, Figure 2 provides a truly global view of how users incur latency due to the root DNS.

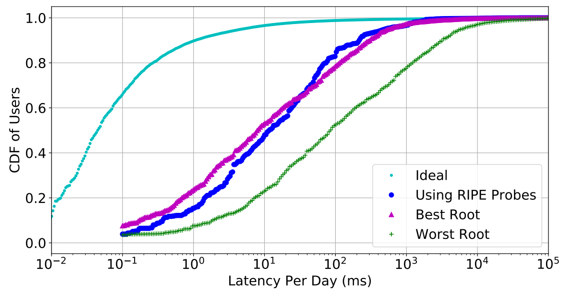


Figure 2: A CDF of approximate latency a user experiences due to root DNS resolution, per day, where the lines represent different ways of estimating root latency. Even using the worst (highest latency) root server, users experience no more than 85 ms per day of root latency.

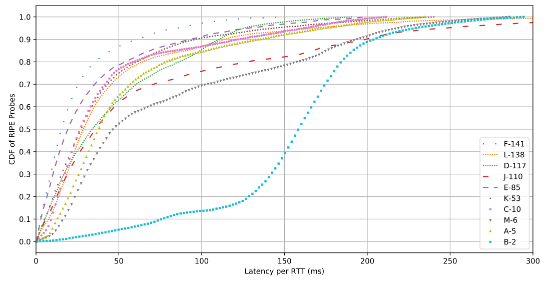


Figure 3: Latencies from approximately 10,000 RIPE probes to root DNS servers on a day in April 2018 (the month of the DITL captures). Letters in the legend are listed in the order of deployment sizes (in April, 2018), and we also show deployment size for clarity. Larger deployments clearly lead to lower latencies, although the trend is not absolute.

Since we do not know the actual latencies from 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. Figure 3 shows a CDF of the median latency over the day, as reported by each of the $\sim$ 10k active observers.

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 (*i.e.*, "leftmost" and "rightmost") in Figure 3. 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 when looking at Figure 3 in isolation. 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 "Using Ripe Probes" looks only at those recursives in the DITL∩CDN data set who are recursives for RIPE probes. We obtained the recursive resolver for approximately 3,000 RIPE probes from collaborators. Our collaborators used 3,000 RIPE Atlas probes (a random subset of RIPE probes are available to execute measurements at any given time) to query DNS records hosted by a server they control and joined logs by the unique DNS record they allotted to each RIPE probe [19]. From Table 1, we see that /24's housing recursives of the above mentioned 3,000 RIPE probes only account for 21% of the volume in DITL and about 0.1% of all /24's; hence, RIPE probes are a small part of the population (as expected). However, for users using these recursives to resolve DNS queries, the 'Using RIPE probes' line could be a fairly accurate measurement of the expected daily latency due to root DNS resolution. Additionally, studies use RIPE probes when measuring anycast latency so this line could provide a useful comparison to prior work [28, 29, 40, 43]. This method provides a low median estimate of 12 ms/day.

Finally, 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.

Since Figure 2 presents daily latencies for each user, it is interesting to put daily latency into perspective. Loading a web page takes roughly 3 seconds on desktop and 7 seconds on a mobile device [9]. Hootsuite, a social media management platform, estimates users spend more than six hours per day on the Internet [34], and another study at USC found American households spend at least three hours per day on the web [27]. Similarweb [54], which tracks usage of various social media services on the web, found Youtube users watch for more than 30 minutes per day on average [54]. Each method of measuring such high-level Internet usage statistics has its drawbacks, but, clearly, latencies in Figure 2 are dwarfed by these statistics. Hence, root DNS latency has little impact on user experience.

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.

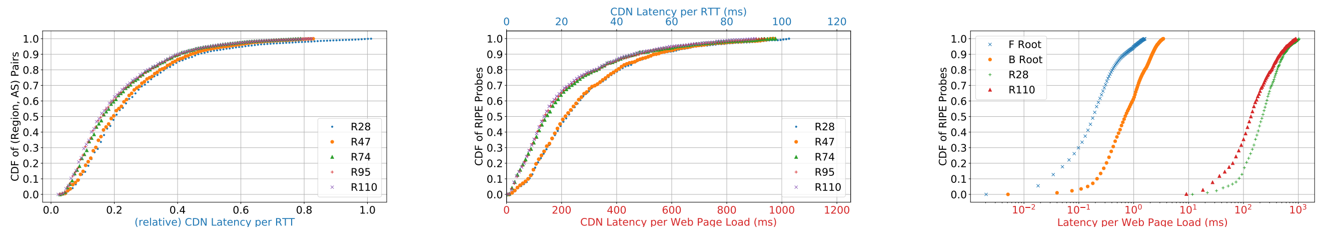
4.2 Impact of CDN Latency on Users

We now measure how users are impacted by latency of a large anycast CDN. Using both client-side measurements and server-side logs, we conclude that anycast latency results in orders of magnitude more visible delay to users for page loads from a CDN than users see due to the root DNS. Consequently, investments in more anycast sites positively affect user experience much more in the case of the anycast CDN.

As discussed in Section 2.4, the anycast CDN we study has *rings* that form a logical hierarchy of layers. Each larger ring adds some sites to those of the smaller ring. Since each ring provides an independent IP anycast CDN service, we report results for each of the rings individually. Rings all use split TCP connections and rely on cloud services for content. As a result, different ring sizes reflect some of the benefit of additional anycast locations, but we caution that they do not reflect all the performance benefits of independent sites.
Users experience latency from the CDN as they retrieve web objects (e.g. web pages or supporting data) hosted by the CDN. Hence, in order to assess how users of the CDN 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.

In Figure 4a, we show latencies from ⟨`region, AS`⟩ locations to rings. We use client-side measurements in Figure 4, as they enable us to fix the set of users for a given ⟨`region, AS`⟩ pair (see §3.2). We show relative latencies for confidentiality reasons. Median latency jumps significantly per RTT (5 ms) going from R47 to R74, but we see minimal changes (2 ms at the median) in global latency for larger rings. These changes in median latency are due to the way in which the front-ends "cover" users of the CDN, a notion that we explore in Figure 5.

Figure 4b demonstrates how RTT latency translates into user-perceived latency in a web page load, for RIPE probes. We use latencies from RIPE probes to quantify user-perceived latency, since we can not share absolute latencies from CDN measurements. Latencies per page load are calculated by multiplying median latencies by the number of RTTs per page load (as in §4.1); for reference, we also include RTT latencies on the top axis. The number of RTTs per page load depends on the size of the page requested and the server's



(a) (b) (c)

Figure 4: RTTs from users to rings as measured by clients (4a), RTTs/latencies per web page load from RIPE probes to rings (4b), and latencies per page load are shown for B root, F root, and R118 on a log scale. Axes with per-RTT latencies are colored blue, while axes with per-page-load latencies are colored red. CDN incurred latency is orders of magnitude higher than root DNS latency (4c).

TCP implementation. We estimate RTTs per page based on the top-1000 pages according to GTMetrix [2] and web pages hosted by the anycast CDN, taking an estimateof 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.

Figure 4b shows users can experience up to 1,000 ms in latency per page load. For large deployments (e.g. R95), half of RIPE probes experience approximately 100 ms of latency per page load. We observed RIPE probes experience lower latencies to this CDN than users do globally (not shown in figure), so Figure 4b likely underestimates the latency users typically experience. Therefore, latency to this CDN therefore clearly factors into user experience.

The difference in median latency per page load experienced by RIPE probes between R28 and R110 is approximately 100 ms, which is a measure of how investments in more front-ends can help users. Although a page load latency reduction of 100 ms may sound inconsequential, prior work has demonstrated that higher latencies can lead to major reductions in numbers of searches [21] (and therefore profits). In addition, *tail latency* has a strong effect on user retention. We show larger rings help tail latency, moving it from about 1,000 ms to 800 ms (Fig. 4b).

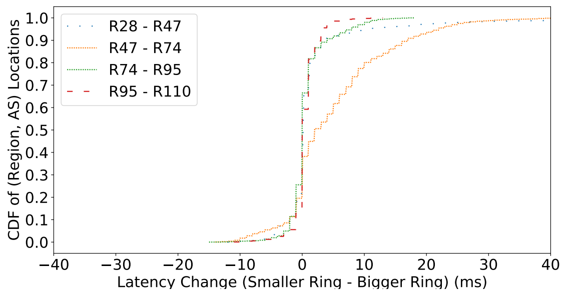
Figure 4 also shows that the incremental performance improvements from more front-ends slow as ring size increases. To explain why these improvements slow, we show changes in (expected) median latency for ⟨`region, AS`⟩ pairs when transitioning to larger rings along with corresponding increases in population "coverage" in Figure 5. Expectation is taken over front-ends with respect to the number of users reaching each front-end, as is done repeatedly throughout the paper.

Figure 5a shows most ⟨`region, AS`⟩ pairs 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. Since Figure 5a uses client-side measurements, we do not know why latencies change when moving from ring to ring, since 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.

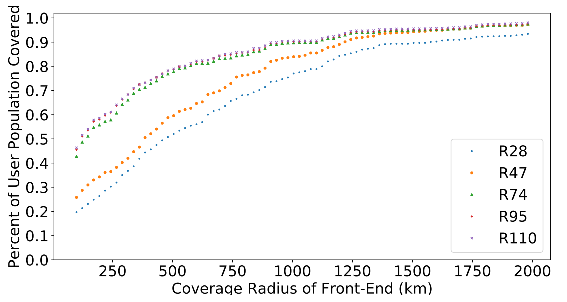
Performance changes are highly correlated with the population "coverage" – the fraction of users that are "close enough" to a site in a ring. To demonstrate this trend, in Figure 5b we plot coverage of each ring as a function of coverage radius. We say a front-end covers a set of users in a ⟨`region, AS`⟩ pair with respect to radius r if the front-end is within r km of the mean location of those users. The intuition is that if a front-end is closer to a set of users, it is likely to offer a low latency option to those users. The latency benefits from transitioning rings correspond well to the increases in population coverage, with the transition from R47 to R74 providing both a large reduction in latency and larger coverage of user population. For example, R74 has a front-end in Madrid, Spain while R47 does not. Users in all regions and ASes of Spain are, at least in part, directed to this front-end when querying R74. However, in R47, users in Spain usually land at a front-end in the United Kingdom. This analysis also provides insight into why diminishing returns are seen as more front-ends are added – eventually, almost all user population is covered.

4.3 Comparison

To compare CDNs with root latency, in Figure 4c we include per page load latencies for F root, B root, R28 and R110. We chose these two root deployments as they provided the best and worst median latency for RIPE probes (see Fig. 3), respectively. Clearly CDN latencies matter *orders of magnitude* more than root DNS latencies, no matter which percentile of which curve we look at in Figure 4c. Additionally, the benefit



(a)



(b)

Figure 5: The change in median latency for ⟨region, AS⟩ pairs when transitioning rings (5a) and the corresponding increasing in population "coverage" of the anycast CDN deployment when adding front ends (5b). Performance generally improves for most ⟨region, AS⟩ pairs when adding more front-ends and the magnitude of the improvement corresponds to how close users are to front-ends.

from using a larger root DNS deployment (F root vs B root) amounts to fractions of a ms per page load, while for the CDN using a larger deployment (R28 vs R110) can make a difference of hundreds of milliseconds per page load.

Looking at the 10^{th} and 90^{th} percentiles of each curve in Figure 4c, we see that the latency penalty for (relative) poor performance among RIPE probes is much greater for users of the anycast CDN. For example, different RIPE probes can experience up to a 200 ms discrepancy in per page load latency when accessing the CDN, while probes only see at most a few ms discrepancy when accessing F root. Since the difference between 10^{th} and 90^{th} percentiles is a measure of the potential return on investment in front-end placement, investments in new anycast sites provide orders of magnitude more benefit for users of the anycast CDN than they do in the root DNS.

Figure 4 is a simple, effective way of demonstrating how users experience latency from the CDN, how much CDN latency “matters”, and how much more it matters to users than root DNS latency. Although the specific quantitative impacts for users may vary by region, AS, operating system, web page, *etc.*, our qualitative observation that CDN latency matters for users, and that it matters a lot more than root DNS latency, is likely insensitive to these variables.

5 COMPARING PATH INFLATION

Having compared latency between the root DNS and an anycast CDN, we now investigate anycast path inflation in each application. As discussed in Section 2.2, anycast may direct users to distant nodes, potentially inflating latencies for users relative to the geographic optimal. In the following we analyze how this inefficiency manifests itself in two different anycasted services, how this inefficiency affects users, and comparisons we can make between root DNS and anycast CDN inefficiency.

Using the DITL captures, we find anycast path inflation in the root DNS can send users to far-away anycast sites, unnecessarily inflating latencies. While this inflation shows latency is not optimal, this inflation *hardly matters* to users, because caching makes queries rare. Conversely, as few as 20% of users of the anycast CDN experience any path inflation, and users experience much less path inflation per RTT than in the root DNS. The CDN’s low inflation suggests it is possible to limit anycast path inflation, and the CDN works hard to control it given the pain it can cause to users.

5.1 Path Inflation in the Root DNS

To measure anycast inflation for the root DNS, we look at how users 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. Notably excluded from the analysis in this subsection are H root, G root and I root, which did not provide non anonymized packet traces at the per-site level.[5]

As discussed in Section 2.2, anycast path inflation is the difference in achieved latency to an anycasted IP and unicast path inflation. Unfortunately, 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 probes) to all sites would stress measurement budgets and rate limits. However, setting unicast path inflation to 0 provides us with an *upper bound* on anycast path inflation. Therefore, using achieved latency as an approximation for anycast path inflation suffices, since we ultimately demonstrate anycast path inflation’s effect on users is quite small.

[5]Anonymization prevents us from seeing where queries originate.

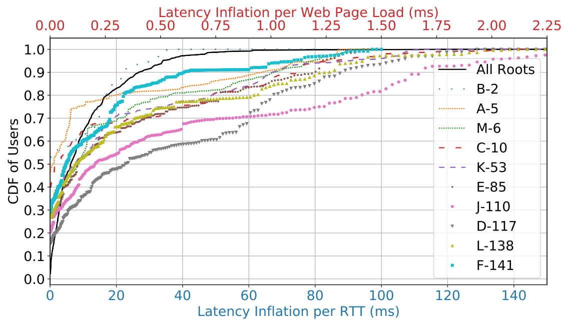


Figure 6: A CDF of latency inflation per RTT (bottom axis) and per page load (top axis) for root deployments, and globally across all roots, experienced by users of a large CDN. Root letters in the legend are listed according to their deployment sizes during the DITL captures, from smallest to largest. Generally, larger deployments are likely to inflate queries. The difference between the best and worst inflation per page load is only about 2 ms in the tail.

Figure 6 shows a CDF of expected inflation per RTT (bottom axis) and per page load (top axis) for users in the DITL∩CDN data set for each letter individually, and globally (across all letters). Each point in Figure 6 uses the following calculation. First we geolocate all recursives in the DITL∩CDN 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 geolocate 0.4% of (valid TLD) DITL query volume in 2018.

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

$$API_j(R) = \frac{2}{c_f}\left(\sum_{i_j} \frac{n_q(i_j)d(R, s_{i_j})}{N_j} - \min_{i_j} d(R, s_{i_j})\right) \qquad (1)$$

where $n_q(i_j)$ is the number of queries to site i by recursive R, N_j is the total number of queries to all sites s_{i_j} 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, $d(R, s_{i_j})$ is the distance between the recursive resolver and site s_{i_j}, and both the summation and minimization are over the sites in this letter deployment. $API_j(R)$ is therefore an approximation of the expected latency 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 an recursive is then the empirical mean over all roots

$$API^g_{root}(R) = \sum_j API_j(R)N_j/N \qquad (2)$$

where N is the total number of queries sent by this recursive. Each recursive is weighted according to the number of (CDN) users behind it, so Figure 6 represents the latency inflation a random user would experience when querying the roots.

It is well known that packets can take circuitous routes and so do not travel great-circle distances to destinations. For example, prior work suggests packets rarely effectively travel faster than $\frac{2}{3}$ the speed of light in fiber ($\frac{4}{9}$ the speed of light) [38], when comparing great-circle distance and measured latency. Hence, our dividing great-circle distance by c_f is a simplification. However, using other constants besides c_f does not change our qualitative conclusions about inflation.

Figure 6 demonstrates the likelihood of a root DNS query experiencing any anycast path inflation (y-axis intercept) roughly grows with deployment size, with a few exceptions. We also find 95% of users experience no more than 40 ms of latency due to inflation when they query the roots. These findings corroborate results presented in prior work [40] that increasing deployment size makes inflation more prevalent.

However, we recommend placing path inflation in perspective in two ways. First, we consider how much actual latency it adds, not just the percentage increase over optimal. Second, we consider all roots and not just inflation from a few specific RIPE Atlas probes to two root letters. As an *absolute cost*, path inflation is almost always under 10 ms for most deployments and most users. Our approximation of anycast path inflation as total inflation is a particularly egregious over-estimation for C root, queries to which are known to exhibit very little anycast path inflation but quite a lot of unicast path inflation [40]. What is perhaps most interesting is that, although users querying any root (line 'All Roots') are quite likely to see non-zero inflation, they see less inflation in the tail than all roots except B root.

As a useful visualization, we also display anycast path inflation *per page load* caused by root DNS queries on the top axis of Figure 6. For this rescaling, we multiply latency inflation per RTT to the root servers by the average number of RTTs to the root servers incurred per page load, the latter of which we denote by f_r. For consistency, we approximate f_r using the statistics found in Section 4.1, as 0.015 (the root cache miss rate of 0.5% times the number of blocking DNS requests per web page). The top axis therefore demonstrates that this inefficiency of anycast does not affect users in any perceptible way. Users querying the root experience less than a millisecond of path inflation per page load, and the difference between the most path inflation (J root) and the least path inflation (B root) is only a couple milliseconds in the tail of per-page-load times.

Looking at Figure 6, one may conclude B root is the most "efficient" deployment, directing users to their closest site most of the time. However, Figure 3 shows that this efficiency comes because B-Root latency is higher than other letters at the time. This latency is in part because B-Root at the time had only sites in the Americas, and a majority of RIPE measurements are from Europe [52]. In early 2020 B-Root deployed three new anycast sites, including sites in

Singapore and Amsterdam, in part to reduce latency to these regions [17]. We hope to redo our numbers with 2020 DITL data to see if the trend changes. However, B-Root emphasizes that inflation does not affect users: Figure 6 shows that the reward for B roots' efficiency amounts to little more than a millisecond when loading a web page.

The above analysis demonstrates the importance of judging the performance of anycast deployments by metrics specific to the applications they serve. Latency inflation and inefficiency in the root DNS is a misleading metric for performance that affects users. It is a poor metric to guide operators, since they have little incentive to fix routing inefficiencies, tweak BGP announcements, or invest in expensive peering or new sites to limit cases of path inflation, because users are not affected. (Additional anycast sites add capacity and spread load to handle Distributed Denial-of-Service attacks [46], perhaps one reason for continued growth in root deployments.)

5.2 Path Inflation in an Anycast CDN

We next investigate anycast performance in a large anycast CDN to quantify inefficiency and anycast path inflation, and discuss their impact on user experience. Comparison to root anycast (Section 5.1) can help identify how design decisions differ by application, and how users of different applications benefit from different choices.

To measure anycast inflation for a large anycast CDN, we use both server-side and client-side measurements (§3.2). Server-side logs are particularly 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.

Recall in Section 5.1, we measured "geographic" path inflation in the root DNS as an approximation of anycast path inflation for queries, where geographic refers to the fact that we really measured distance inflation of queries. The anycast CDN measures actual latency from users to front-ends, allowing a more precise estimate of both anycast latency inflation and geographic inflation than we can obtain for the root DNS. 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`⟩ are representative of all users in that ⟨`region, AS`⟩ (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. This occasional rerouting allows performance comparisons across sites for the same set of users.

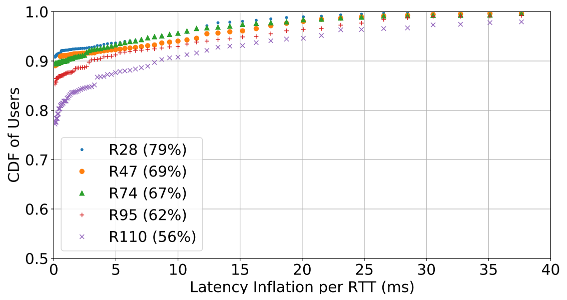
In Figure 7a and Figure 7b, we show the two aforementioned ways of measuring anycast path inflation for users of the anycast CDN: actual anycast path inflation and geographic path inflation. For each ring, we calculate actual anycast path inflation, $API^{A}_{CDN}(r, a)$ for users of region r and AS a as

$$API^{A}_{CDN}(r, a) = max(0, \sum_{i} \frac{l_i N_i}{N} - l_{j^r}) \tag{3}$$

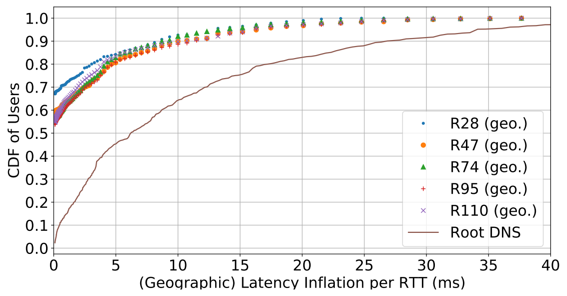
where N_i is the number of users in (r,a) that visited front-end i, N is the total number of users in (r,a), l_i is the median latency of users in (r,a) towards front-end i, and front-end j^r is the closest front-end to r. We calculate geographic path inflation in Equation (1). Unlike prior work, we compare latencies to the closest front-end, rather than the best performing front-end [22, 40], since we do not have measurements from every ⟨`region, AS`⟩ pair to every front-end, so we do not know the best performing front-end. The difference in Equation (3) can be negative, as routes to geographically close front-ends may be circuitous, so we treat negative differences as zero. Some ⟨`region, AS`⟩ pairs do not visit their closest front-end, and so each line in Figure 7a represents a fraction of users in that ring – from R28 to R110 between 79% and 56% of users are represented (shown in legend). We explore analysis that looks at the same set of users across rings in Appendix A.5, reaching similar conclusions.

As with root DNS, we see greater anycast path inflation as the number of front-ends grows (Fig. 7a). For example, in R28, fewer than 10% of users experience any inflation, while in R110, more than 20% of users experience some inflation. However, larger rings generally provide users lower absolute latency, as demonstrated in Figure 4 — inflation shows greater inefficiency, but absolute performance improves.

CDN users usually experience no geographic latency inflation (Fig. 7b, y-axis intercepts), and users are seldom directed to far-away nodes, with 80% of users experiencing less than 5 ms (approximately 500 km) of geographic inflation per RTT for all CDN rings. Conversely, users of the root DNS usually 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 the CDN inflation per RTT (at every percentile) suggests the CDN works harder to control it, perhaps through peering directly with user ("eyeball") ASes, and with active debugging of bad routes.The root reason for these differences may be that, since users see minimal practical benefit from lower latency, root DNS operators have little incentive to reduce inflation. By contrast, latency is more important in an anycast CDN, with strong influence on user experience and CDN profits, and reducing inflation is an opportunity to improve.



(a)



(b)

Figure 7: Anycast path inflation measured using CDN server side logs (7a) and using geographic information (7b). Path inflation becomes more prevalent for larger deployments, but it is still quite small for most users. Path inflation in the anycast CDN is also much less common than in the root DNS.

5.3 Comparison

Having presented the analysis of anycast for two different services, we now take a step back and summarize the key differences in efficiency. One problem anycast introduces, path inflation, is much lower for the anycast CDN than in the roots. Not only do users tend to go to closer CDN sites than they do root DNS sites, but these paths are also rarely inflated at all. This result is not surprising, when we consider how much more latency matters in a web page load, than does root DNS inflation. Although root DNS queries see greater inflation (and see inflation more often), when inflation does occur for CDN queries, it hurts user-experienced latency (here, page load-time) much more than equivalent inflation for DNS root queries. Combined, these observations suggest the CDN is working harder to control cases of sub-optimal routing, since users feel the pain.

Another way of measuring inefficiency in routing is to look at where queries are being routed with respect to users generating those queries, without paying mind to incurred latency. Observing that most users reach their closest or second closest site suggests an anycast deployment is 'efficient'. We explore this notion further in Appendix E. However, geography is not a perfect indicator of efficiency. An assessment of anycast routing inefficiency based solely on geography neglects geopolitical restrictions, physical restrictions (e.g. topography), and routing constraints.

6 DISCUSSION

Anycast performance is interesting to assess in its own right, even without attention to how that performance affects end users. However, the magnitude of *problems* caused by anycast's inefficiencies are proportional to their effect on end users. Hence, conclusions we draw about "what should be done" in the face of such inefficiencies depend on the service the anycast is providing. This principle extends beyond anycast and should be applied to any system.

Recent work has suggested that anycast inefficiency (path inflation) is a serious problem, based on analysis of root DNS deployments [40]. The authors argue that adding more sites may hurt rather than help deployments (due to inflation), and suggest 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 an additional type of anycast deployment – 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.

Each letter root and the CDN are run by different organizations and so have different operational budgets, deployment strategies, and peering/routing strategies. These differences mean that comparisons between, for example, D root and F root may be uncontrolled, in the sense that they vary in multiple dimensions outside the one of interest. For example, Figure 6 might suggest that more sites leads to more inflation. However, asserting this ignores the fundamental differences among the root letter deployments.

Our analysis of CDN data provides evidence that network operators can control anycasts' inefficiencies, even though these inefficiencies are quite prevalent in the root DNS. Moreover, inefficiency does not tell a complete story, since systems such as the root DNS do 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 the CDN 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 for services, including the root DNS.

7 RELATED WORK

IP anycast performance is usually studied in the context of two applications: the root DNS servers, and CDNs. In addition to these topics, we discuss studies of popular recursives, and user-centric measurements of web performance.

7.1 Root DNS Anycast

The performance of anycast in the context of root DNS is generally gauged by anycast's ability to balance load among server sites or provide low latency to users. Previous work looked at DDoS attacks on the root name server infrastructure, and showed that anycast is a helpful defense mechanism against such attacks [46, 51].

Other work looked at anycast performance and found it is decent for most roots [51] but is related to good geographic locations of anycast sites and peering strategies [29]. These findings coincide with an earlier study that concluded the performance of anycast is intrinsically linked to deployment strategy [18]. Towards assessing how size of the deployment correlates to performance, previous work suggested that as few as 12 sites can provide "good" latency to users [29].

A final line of work looked at the *efficiency* of anycast latency, suggesting that more sites decrease efficiency [40]. Many studies quantified latencies to various root servers, and noted how these compare to the (optimal) latency of the closest unicast alternative for the user who issued the query [28, 29, 41].

Our paper focuses on performance, considering latency and efficiency, and does not consider DDoS mitigation. We are the first to examine two different applications to see how the importance of latency drives CDNs to greater efficiency. Finally, we show that efficiency is less important than absolute latency, suggesting that emphasis on inefficiency in larger deployments is perhaps misplaced.

7.2 CDN Anycast

Some CDNs [15, 22, 55] use IP anycast to augment their serving infrastructure. The simplicity of IP anycast for CDNs comes at the cost of having coarse grained control over where user queries land; for example, shifting user load between nodes during peak hours is a challenging problem. Some CDNs use DNS redirects at ADNS servers to shift load among anycast nodes \cite{alzoubi2011practical, {flavel2015fastroute}. Previous work analyzed what latency CDN users are achieving, compared to optimal, when being routed to anycast nodes and found that 10% of users experience a latency inflation of at least 100 ms [22]. Our work builds on this result by instead focusing on how the size of the deployment affects user experience.

7.3 Recursive Resolvers and the Benefits of Caching and Latency

Similar to the recursive analysis conducted here, others have looked at DNS traffic on a small network and found that 16% of queries resulted in queries to the root, most of which were for invalid domains [36]. As this study is quite old, it is no surprise that this rate has decreased (recall we observed 1.5% of queries resulted in queries to the root) since browser designers and network engineers understand the importance of caching. Previous work also looked at traffic between a neighborhood and a recursive and analyzed statistics of DNS exchanges occurring over it including DNS transaction latencies [24]. Some previous work looked at certain pathological behaviors of popular recursives, and the implications these behaviors have on root DNS load times [39, 58].

Work has examined recursive selection across multiple authoratives, showing the extent to which preferences for lower latency shift traffic [47]. Our work looks at how authoritative anycast systems may optimize latency.

7.4 Web Performance

Many studies characterize web performance and consider DNS, although none consider root DNS specifically. Researchers have characterized web performance bottlenecks in (at the time) new broadband networks, and found that latency was the main bottleneck for page load time when the user's bandwidth exceeds 16 Mb/s [53]. However, the study did not realistically simulate a page load and did not analyze the effect of having multiple DNS resolutions per page. Others analyzed how each step of a page load contributes to the aggregate PLT using a tool designed in-house [16]. This work, however, did not conduct a large measurement campaign and did not include information about multiple DNS lookups per page. Finally, HTTP Archive [9] is an ongoing effort to characterize web performance, but only measures from a single vantage point.

8 CONCLUSION

Anycast is widely used in many systems to provide content to users, but it has come under fire for routing users to suboptimal sites. Research usually uses the root DNS to demonstrate this suboptimality, but users rarely interact with the root DNS since caching is so effective. Taking a user-centric approach to studying anycast performance, we show that root DNS performance doesn't matter for users and that for anycast CDNs, performance can be quite good. Although inefficiencies do exist, anycast still does a good job routing users to sites.

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. Amazon Route 53 FAQs. aws.amazon.com/route53/faqs/

[7] 2020. Data center Locations. opendns.com/data-center-locations/

[8] 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

[9] 2020. The HTTP Archive Project. httparchive.org/

[10] 2020. IANA IPv4 Special-Purpose Address Registry. iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

[11] 2020. Root Servers. root-servers.org

[12] 2020. Tshark. wireshark.org/docs/man-pages/tshark.html

[13] 2020. Window: load event. developer.mozilla.org/en-US/docs/Web/API/Window/load_event

[14] 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

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

[16] 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.

[17] B-Root. 2020. B-Root Begins Service from Singapore, Virginia, and Amsterdam. blog post https://b.root-servers.org/news/2020/01/28/new-sites.html. https://b.root-servers.org/news/2020/01/28/new-sites.html

[18] 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.

[19] Lucas Barsand, Artur Vaz, João Bastos, Osvaldo Luis Fonseca, and Italo Cunha. 2019. Realizando o Potencial da Plaforma RIPE Atlas. In *Salão de Ferramentas do SBRC 2019*.

[20] Ray Bellis. 2015. Researching F-root Anycast Placement Using RIPE Atlas. ripe blog https://labs.ripe.net/Members/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas. https://labs.ripe.net/Members/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas

[21] Jake Brutlag. 2009. Speed matters for Google web search.

[22] 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.

[23] 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.

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

[25] 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

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

[27] Jeffrey Cole. 2018. Surveying the Digital Future. digitalcenter.org/wp-content/uploads/2018/12/2018-Digital-Future-Report.pdf

[28] 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).

[29] 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.

[30] 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.

[31] 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.

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

[33] 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

[34] Hootsuite. 2020. Digital 2020. hootsuite.com/resources/digital-2020

[35] Geoff Huston. 2014. How Big is that Network.

[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] 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.

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

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

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

[46] 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.

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

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

[53] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. 2013. Web performance bottlenecks in broadband access networks. *ACM SIGMETRICS Performance Evaluation Review* 41, 1 (2013), 383–384.

[54] Pavel Tuchinsky. 2016. Which Social Media Apps are Going Up? similarweb.com/corp/blog/which-social-apps-are-rising/

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

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

[57] 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.

[58] 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.

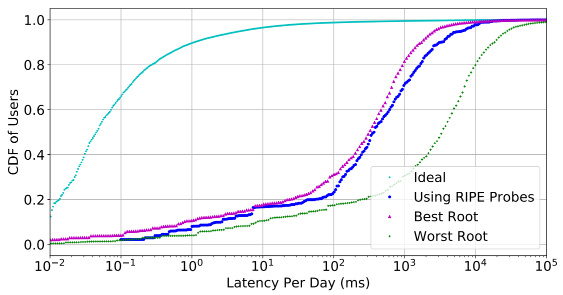


Figure 8: Daily latency experienced by users due to the root DNS, calculated by amortizing root DNS requests over user populations, when including or excluding queries for invalid TLDs. Counting invalid queries drastically increases daily user latency estimates to 327 ms for Best Root (a 30-fold increase).

A QUANTIFYING THE IMPACT OF METHODOLOGY DECISIONS

When analyzing latency and inflation, we often make assumptions or choose to conduct analysis a certain way. In what follows, we justify our various assumptions and preprocessing steps, analyze the effects of these assumptions on our results, and, when possible, offer alternative methods of performing the analysis.

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, increasing 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 [14]. Researchers at USC have found that more than 90% of single-label queries at the root match the Chromium captive-portal pattern [32]. 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 [30]. It is unlikely these queries are caused by typos, since they are actual (as opposed to misspelled) words and resemble domains often seen in software or in corporate networks. Chromium queries and queries for a certain set of invalid TLDs therefore account for around 86% of all queries for invalid TLDs at the roots, suggesting the vast majority of queries we exclude are not directly associated with user latency.

Nevertheless, it is still valuable to assess how including these queries for invalid TLDs changes the conclusions we can make about root DNS latency experienced by users. Figure 8 shows daily user latencies due to root DNS resolution when we include requests for invalid TLDs and PTR records in daily query volumes. Using 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 8 is the number of users (not /24s), counting invalid queries shifts the graph far to the right. Hence, counting invalid queries drastically affects the conclusions we can draw. 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 [9]), 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 a large CDN who use recursives (§3.1). Naturally recursives used by users of the large CDN and recursives seen in DITL do not overlap perfectly. To increase the representativeness of our analysis, we aggregate CDN user counts and DITL query volumes by resolver /24, and join the two data sets on /24 to create the DITL∩CDN data set. The intuition behind this preprocessing step is that

Data Set	Statistic	Percent Overlap
DITL ∩ CDN	DITL Recursives	2.45% (29.3%) of DITL Recursives
	DITL Volume	8.4% (72.2%) of DITL Query Volume
	CDN Recursives	41.9% (78.8%) of CDN Recursives
	CDN Volume	47.05% (88.1%) of CDN Query Volume
DITL ∩ CDN ∩ RIPE	DITL Recursives	.07% (.14%) of DITL Recursives
	DITL Volume	11.8% (20.7%) of DITL Query Volume
	CDN Recursives	.01% (.34%) of CDN Recursives
	CDN Volume	.62% (54.6%) of CDN Query Volume

Table 3: Statistics displaying the extent to which the recursives of users in a large CDN overlap recursives seen in the 2018 DITL captures without users and volumes by /24. Also shown in parentheses are corresponding statistics when joining by /24. Joining the data sets by /24 increases most measures of representation by tens of percents, with some measures increased by up to 64%.

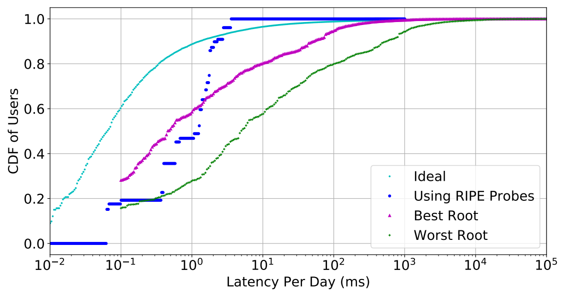


Figure 9: A CDF of approximate latency a user experiences due to root DNS resolution, per day, without joining recursives in DITL with recursives seen by the CDN by /24. This unrepresentative analysis yields an estimate of daily user latency far lower than in Section 4.1.

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 3 we summarize the extent to which the recursives seen by the CDN are representative of the recursives seen in DITL, and vice-versa, without aggregating by /24. We also display corresponding statistics (from Table 1) 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 9 we show daily latency experienced by users of a large CDN 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 3).

Table 3 and Figure 9, demonstrate that the decision to aggregate statistics and join DITL captures with CDN 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 [56], and so we expect that, over the course of DITL, IP addresses in the same /24 reach the same anycast sites.

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

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

where j_F is the favorite site for /24 k (*i.e.,* the site the /24 queries the most), and Q^k is the total number of queries from /24 k. We plot these fractions for all /24s in DITL, and for each root deployment.[6]

For more than 80% of /24s, all queries visit only one site per root letter, suggesting that queries from the same /24 are routed similarly. This analysis is slightly biased by the size of the root deployment. For example, two IP addresses selected at random querying B-root would hit the same site half the time, on average. However, even for L root, with 138 sites, more than 90% of /24s direct all queries to the most popular site. We believe Figure 10 provides evidence that recursives 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 generated by an IP address to a root be a point mass, with all the queries concentrated at that IP addresse's 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 this decision is justified, since it corroborates our results regarding daily user root latency using very different methodology. Specifically,

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

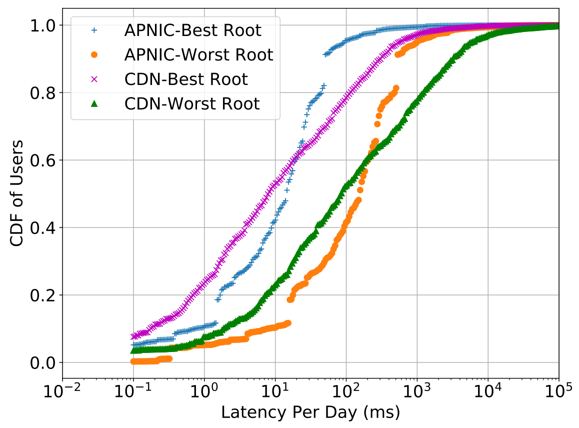


Figure 11: Daily latency experienced by users due to the root DNS, calculated by amortizing root DNS requests over user populations. The APNIC and CDN lines use two different methods of estimating user counts behind each, but arrive at the same basic conclusion – users do not experience much latency due to the root DNS.

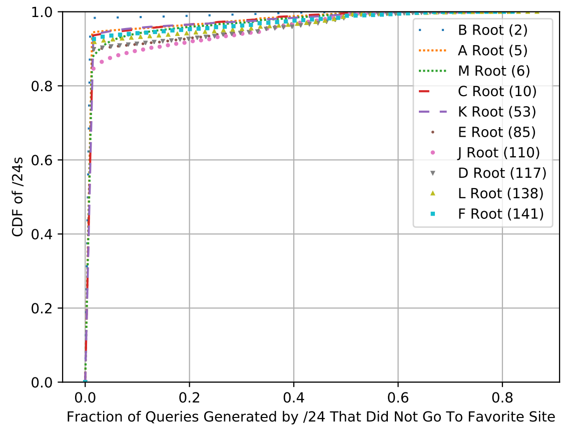


Figure 10: Fractions of queries generated by /24s that do not hit the most popular site for each /24 and for each root letter in question. For all root letters, more than 80% of /24s have all queries visit the most popular site, suggesting queries from the same /24 are routed similarly.

without relying on aggregating statistics by /24 or using CDN 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 instead of CDN user counts to amortize root DNS latency [35]. 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 to an ASN seen in DITL, 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 11 for Worst Root and Best Root, and include the corresponding lines using CDN 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 the large CDN. We provide an estimate of the number of RTTs based on modeling and evaluation of a set of 1,000 web pages using Selenium (a headless web browser).

CDN 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

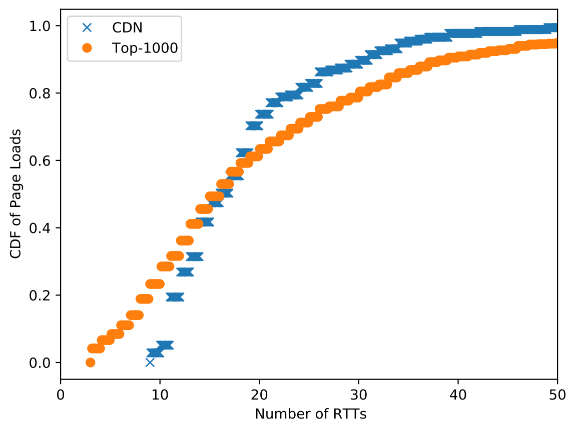


Figure 13: The number of RTTs for page loads. CDN and Top-1000 refer to the types of pages being loaded. Only a few percent of CDN web pages are loaded within 10 RTTs and 90% of all page loads are loaded within 20 RTTs.

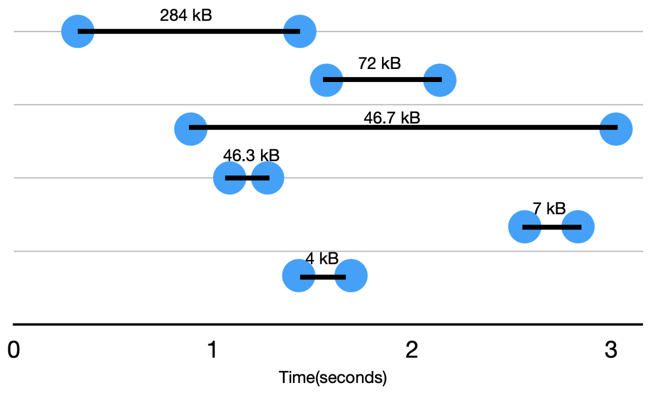


Figure 12: Example TCP connections for a web page loading. Each line is a TCP connection. Disregarding smaller downloads that overlap longer ones, we choose to accumulate RTTs associated with the 284 kB, 72 kB and 7 kB downloads.

$$N = \lceil log_2 \frac{D}{W} \rceil \tag{5}$$

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 [25, 33]. Although W is set by the server, a majority of web pages set this value to approximately 15 kB [50], so we use this value. (We do not consider QUIC in detail here, but with its larger initial window it will see fewer RTTs.)

In Section 4.3 we aim to show CDN 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 **TBD: cite**, 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 which 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 12. 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 the large anycast CDN that we study in this paper (twenty times for each page). We use Selenium and Chrome to open web pages and use Tshark [12] 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 [13]. 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 (5), and add a final two RTTs for TCP and TLS handshakes.

The RTT estimates are shown in Figure 13. As Figure 13 demonstrates, only a few percent of CDN 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.

A.5 CDN Path Inflation Over the Same Set of Users

In Figure 7 we calculate path inflation for users by subtracting latency to users' geographically closest front-ends from user's achieved latency. However, some ⟨`region, AS`⟩ pairs do not reach their closest front-end, so we can not calculate

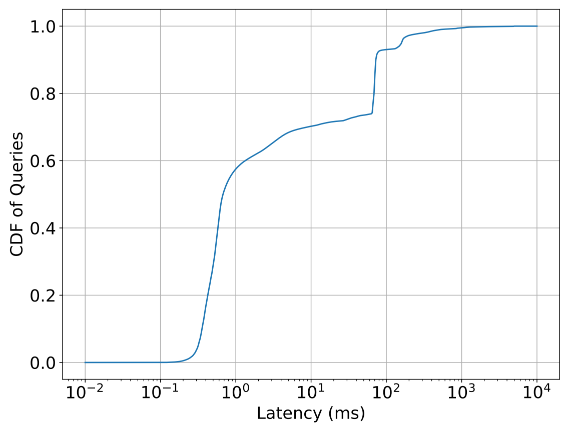


Figure 15: CDF of user DNS query latencies seen at a recursive resolve at ISI, over the course of one year. Latencies are measured from the timestamp when the recursive resolver receives a client query to the timestamp when the recursive sends a response to that client query. The sub-millisecond latency for more than half of queries suggests most queries to this recursive are served by the local cache.

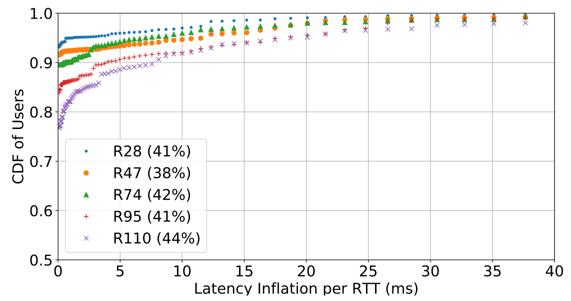


Figure 14: Anycast path inflation measured over rings for the same set of users. Path inflation is slightly less prevalent for this set of users than all users for which we can measure inflation (Fig. 7). This result makes sense, since we are excluding users that do not visit their closest front-end, and so are inflated.

path inflation for these users. This set of users for which we can calculate anycast path inflation varies across rings.

We now calculate anycast path inflation for only those ⟨`region, AS`⟩ pairs that have latency measurements to their closest front-end in all rings. Limiting the analysis to a certain set of users allows a direct comparison in inflation over the same set of users. The resulting path inflations, and user representations (as a fraction of total users in that ring) are shown in Figure 14. Path inflation is slightly less prevalent for this set of users than all users for which we can measure inflation (Fig. 7). This result makes sense, since we are excluding users that do not visit their closest front-end, and so are inflated.

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 15 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 [24]. 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 15. To visualize just how small this fraction is, Figure 16 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 16 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

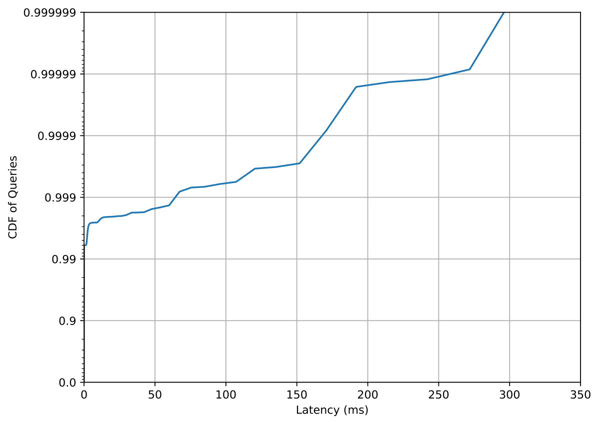


Figure 16: Root DNS latency for queries made by users of the research lab recursive resolver during 2018. This plot demonstrates the benefits of crowd-sourced caching and high TTLs of TLD records – fewer than 1% of queries generate a root request, and fewer than .1% incur latencies greater than 100ms. User queries that did not generate a query to a root server were given a latency of 0.

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 4: 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.

can be considered complementary since we discover different reasons for redundant requests [30].

To observe these redundant requests in a controlled environment, we deploy a BIND instance (the resolver 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 [12].

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 4.

In Table 4, 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 4, 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.

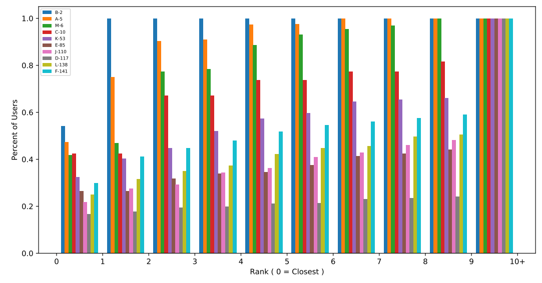


Figure 18: A cumulative histogram showing whether users tend to hit the closest available site for each root deployment. Root letters in the legend are listed according to their deployment sizes at the time, from smallest to largest. Generally, with increasing deployment size, there is an increase in "inefficiency" – fewer users hit closer sites.

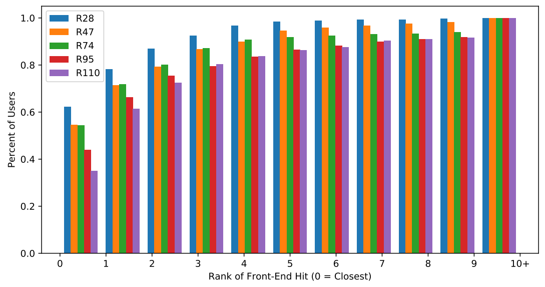


Figure 19: A cumulative histogram showing whether users tend to hit the closest available site for each ring. With increasing deployment size, there is an increase in "inefficiency" – fewer users hit closer sites.

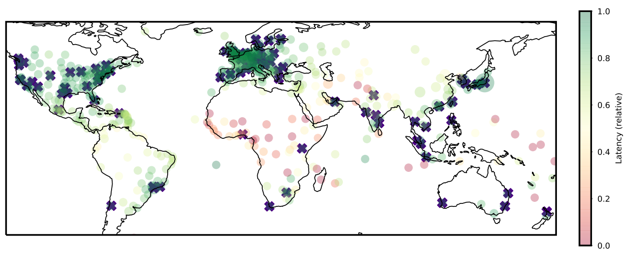


Figure 17: A visualization of front-ends in R110 (purple Xs), and user populations (transparent circles). User populations are colored according to the relative latency they see, and have size proportional to user population. Red corresponds to high latency and green corresponds to low latency. Latency generally gets lower, the closer users are to a front-end, and front-ends are concentrated around large user populations.

D ANYCAST CDN PERFORMANCE VISUALIZATION

In Section 2.4 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 17 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.

E SUPPLEMENTARY ANALYSIS OF ROUTING EFFICIENCY

As a way of measuring how efficient the root DNS and CDN route users to queries, and how efficiency changes among root letters and rings, we show how users are routed to sites. Section 5.1 and Section 5.2 show that path inflation either does not matter to users, or is trumped by performance improvements from more sites. So, looking at inefficiency is interesting but does not fully describe how anycast performance affects users.

The cumulative histogram in Figure 18 provides us with a measure of the inefficiency that accumulates when adding more anycast sites. In Figure 18, the bar for root j and rank i measures the number of users hitting no further than the i^{th} closest site in deployment j. For the purposes of calculation, if N_R users use recursive resolver R, and $q_i^j(R)$ is the fraction of queries from recursive resolver R hitting no further than the i^{th} closest site in root deployment j to that recursive, we accumulate $N_R q_i^j$ users to rank index i for root deployment j (*i.e.*, for the purposes of weighting).

One might expect that as the number of anycast sites increases, the likelihood that a user hits the closest site decreases, and Figure 18 roughly demonstrates this trend. However, there are noticeable exceptions to this rule – roughly twice as many users reach the closest or second-closest site when querying F root compared to when querying D root, despite the fact that F root has 30 more anycast sites. Exceptions to the trend shows the success of some root anycast operators at identifying and correcting anycast routing problems [20].

As a comparison to Figure 18 in Figure 19 we show how "inefficient" each ring is. As in the root DNS, as deployment size increases, fewer users go to their closest sites (as more relatively close competitor sites are deployed) and inefficiency increases. In the root DNS, this inefficiency was inconsequential because users rarely queried the roots. For the anycast CDN, users do interact with the anycast CDN quite a lot (as shown in Section 4.2); however, as demonstrated in Figure 5 it is possible to limit the impact of this inefficiency on users.