If a Path is Inflated, and Noone Uses It, Is It Inefficient[a]?

Thomas Koch (Columbia University)
Matt Calder (Microsoft Research)
Ethan Katz-Bassett (Columbia University)
John Heidemann (ISI)
Arpit Gupta (UCSB)

ABSTRACT

Anycast is a means of distributing content that has been praised for its simplicity and performance, yet criticized for inflating user latencies in some cases. Studies of anycast often use the Root DNS System as a subject, since it is publicly documented, relatively large, and DNS is an important part of the Internet. The implication of studies of Root DNS anycast is that inefficency due to path inflation results in large latency to users. Here we show that, while Root DNS Anycast does have some path inflation, the *actual effects of any inflation to users is tiny*—each user sees on the order of 2,ms per day, far less than the time to unsuspend a laptop computer. Users see little effect of even apparently large path inflation because DNS (in general), and specifically the Root DNS contents, are heavily cached. We suggest that, while the Root DNS system is important and a valuable target of study, latency should not be the motivation.

1 INTRODUCTION

IP anycast, a system in which geographically diverse servers known as anycast replicas all use the same IP address, is an important part of a number of operational DNS [1, 2, 7, 13, 20] and CDN [c][8, 11, 34] systems today, in part because of its support to improve latency to clients and decrease load on each anycast server [22, 28, 31]. However, numerous studies have argued that anycast provides sub-optimal performance for some users, compared to the lowest latency anycast could offer in theory. This sub-optimality implies a great latency cost to the user. Despite these implications, how these inefficiencies impact user experience is not well understood. [d]

Anycast is primarily deployed in two domains: DNS and CDNs. The root DNS servers feature in studies involving anycast because it is relatively easy to gain access to root DNS data, because it is relatively easy to gain access to information about their deployments, and because they are run by several organizations [1]. This last fact manifests itself in a diverse set of deployment strategies, for essentially the same service. For example, in 2018 B-root had 2 server deployments while K-root had close to 70. These different strategies allow researchers to, for example, coarsely analyze the effect the number of sites has on the performance of the service [15], [30], [16], [25], [27].

The geographic diversity of anycast deployments offers resilience to network outages and targeted attacks [25], [30]. However some studies suggest that adding more sites may harm, rather than help, user performance [25], [32]. These studies suggest that BGP can sometimes direct users to far-away anycast servers, despite the existence of a geographically close server. Studies have framed the problem as a sort of optimization problem, where every user request should be mapped to the same physical site as the best unicast alternative [27], [25], [16]. However it is not immediately clear that anycast's problems and inefficiencies, i.e. deviation from the optimal, lead to a measurable impact on user performance, in the context of root DNS resolution.

We argue that using the root DNS servers to draw conclusions about the performance of anycast as a whole has its drawbacks[e], especially when these conclusions are applied to anycast CDNs. Organizations in charge of managing these services (DNS servers and CDNs) have different performance goals when deploying new servers, and the types of content they serve are quite different. For example, generally, new root server deployments offer more resilience in the face of attacks on the DNS infrastructure and so organizations may opt to place their servers in locations that maximize reachability to at least one replica in the face of server failure [30]. In addition, root servers serve a small amount of (static) content, and so there is much opportunity for caching across a large user population. Conversely, caching content is more difficult since a CDN serves such a wide variety of content. Additionally, managers of large content delivery networks (CDN's) may opt to place their servers near large population centers to provide low latency access to users and invest in expensive peering with edge networks. Indeed the everyday notions of "performance" that typical users are interested in are principally important to CDN's, yet may only be somewhat important to root DNS managers.

Our primary goal is to contextualize research[f] results regarding the performance of anycast in the context of the root DNS servers. Our primary goal is to demonstrate that concerns raised about root latency inflation by root DNS servers are exaggerated, given the tremendous capability for caching at the recursive servers. Since there is this tremendous caching capability for root records in particular, we believe anycast's suboptimal performance in the context of the root DNS servers translates to a negligible effect at the end user.
We first look at a recursive resolver to understand root traffic (section 3). We find that, in our limited case study of a single resolver, no more than INSERT_NUMBER percent of a page load is spent waiting for root DNS resolution. Such a small percentage suggests even amortizing root latency over small numbers of users drastically limits user exposure to the root DNS infrastructure. We then leverage the Day in the Life of the Internet (DITL) [3] packet captures to estimate the daily latency experienced by users around the world (section 4). We find that users around the world experience a median latency of INSERT_NUMBER ms of latency due to DNS resolution per day, with a median number of INSERT_NUMBER requests per day. These figures are negligible and confirm our suspicions that, globally, caching makes querying the root DNS server a rare experience. **TODO**: We then investigate querying behavior of recursive resolver software, and discuss the causes and effects of inefficient root querying behavior (subsection 3.3). We present evidence that much of these unnecessary queries are due to flaws in recursive resolver design. Finally, towards steering the future work regarding the root DNS servers in a more fruitful direction, we demonstrate heuristically that focusing on limiting the number of unnecessary queries to the roots results in improved user performance orders of magnitude greater than eliminating anycast path inflation (**??**).

2 DNS – A PRACTICAL VIEWPOINT

The DNS ecosystem has evolved towards caching DNS records closer and closer to the user, and this design principle generally results in the reduced latency for the user. Herein we describe the implementation of DNS, both user and server side, and discuss how this allows for faster DNS resolution at the user.

2.1 Great Latency Savings Potential

There are many descriptions of DNS [23], [14], and we don't attempt to give an introductory lesson. The DNS is a means by which user requests for human-readable hostnames are mapped to IP addresses. Typically, a user will send DNS requests in the form of UDP packets to one or more recursive resolvers (RR's) provided by their ISP[1]. The RR then requests the records from a root DNS server, top level domain (TLD) server and authoritative DNS (ADNS) server corresponding to the record the user requested. Since each request is a correspondence between the RR and a remote server, there can be several requests made by the RR for a single end-user request.

In practice, there is quite a lot of potential for caching these DNS records, thus removing unnecessary steps of the recursion outlined above. Each DNS server provides a time-to-live (TTL) with each record they return, specifying the duration in seconds for which the record can be kept locally cached. After this duration, the record should be deleted from the cache of the RR [29]. It has been observed that more heavily accessed websites tend to purposefully set the TTL of their DNS records to be some small value, evidence of fine-grained traffic engineering [12]. Hence the popularity of a website does not make it more likely to live in the cache of an RR. Nevertheless, RR's can pre-fetch records for popular websites. For example, a configuration setting on the popular resolver BIND specifies whether or not BIND should prefetch (recently accessed) records set to expire soon [19]. In contrast to heavily accessed sites, TLD records tend to have long TTL's. For example, the COM TLD record (one of the most popular) has a TTL of 2 days. One might speculate that, since the most popular websites fall into ten or so popular TLD's [4], that a request to the root server would rarely occur. In this scenario, a user fetching example.com would only generate a request for, say, the COM NS record once every 2 days. Furthermore, some commercial RR's are

[1] The user can specify whatever RR they wish, but one can sensibly assume the typical user's RR is set by the ISP, broadcasted through DHCP.

known to cache the entire root table locally. Since there are only 1,000 or so TLD's, the memory requirements are quite inexpensive and caching these records provides potential for latency savings. [5] goes one step further to suggest every RR should have a local copy of the root. On top of all of this structure, there can be multiple RR's (a sort of hierarchy of RR's) among which a query traverses before a third party (root, TLD, ADNS) is finally queried. This hierarchy compounds the potential effects of local caching.

Caching at remote RR's is wonderful, and can greatly reduce DNS query resolution time. Indeed since the typical user's RR is run by their *local* ISP, generally the user will experience negligible latency for a DNS query if they hit a warm cache. Modern browsers and operating systems have taken this one step further and engage in a number of practices that further aid the user. Operating systems or browsers are known to locally cache DNS records, as these records are small (compared to say, an image). Software can go even further, refreshing records in the cache that are about to expire (just as a RR can do). Features such as link prefetching in Chrome and Firefox, where the browser will send DNS requests for all links on a page, allow the user to completely "skip" the DNS resolution process in certain scenarios. All this being said, it is unfortunately quite difficult to quantify the extent to which these features save users time. Users request web services through an idiosyncratic combination of browsers, web sites, operating systems, personalized settings [2], physical locations, bandwidths, devices, ISP's, and RR's. On top of this, all the optimizations just mentioned can be undone through poor configuration, a conservative manager of the RR, or software inefficiencies as we will now discuss.

2.2 Practical DNS Pitfalls

We have discussed the ways in which a user is, by design, prevented from experiencing latency due to root DNS query resolution. However, there can be a stark difference between ideal and actual caching behavior. Although RR's should cache TLD records for the full TTL, unexpected logical flows in software execution can cause unnecessary queries to the root server. Furthermore, operators can set conservative settings on the RR's they run. Regardless of the TTL recommended by the record, the operator can set this to any value below that recommended TTL. It has also been noted that popular RR distributions have implementation flaws that lead them to select poor/unresponsive DNS servers as opposed to low-latency, high performing ones [35]. Furthering the problem, users can disable local DNS caching on their computers, or use services such as private browsing which disables some local caching for privacy concerns. Web pages can also contain "hidden" DNS requests. The HTTP body of a web page can contain references to resources stored on third party domains, for which the browser must request DNS records. Those resources can contain even more references, creating a chain of DNS requests for potentially uncached records. Another source of root DNS latency, reverse-DNS lookups or requesting invalid domains always result in queries to a root server. Users can also act as their own RR if they wish, eliminating much of their ability to leverage the full benefits of caching that are only possible through the "crowd-sourced" caching of ISP or regional RR's.

It is difficult to quantify or measure the degree to which these pitfalls limit a typical users' ability to leverage the caching capability of DNS. Notably it is difficult to characterize the inefficiency introduced by the software employed by the RR of that users' ISP. One could potentially statistically characterize these inefficiencies for various open source RR's, which is a subject for future work.

3 A CLOSE LOOK AT A RECURSIVE RESOLVER

Towards the goal of quantifying the extent to which latency due to root DNS query resolution impacts the typical user at a particular institution, we analyze packet traces of a recursive resolver at ISI. Although this is a fine-grained analysis, looking at a single resolver allows us to estimate the average amount of latency experienced by a user due to DNS resolution *per page load.* This type of analysis would not be possible from a single end user, nor from the vantage point of the root servers themselves.

3.1 Data Source

The recursive resolver of interest (running BIND 9) saves all traffic traversing over port 53 to file, and has done so for 5 years, providing us with a rich source of data. This corpus of users is relatively small, and consists of university traffic, so the specifics of the analysis we conduct may not extend to other RR's. For example, over 2018, we saw roughly 900 unique IP addresses, with about 100 unique IP addresses each day. This might be a smaller corpus of users than is seen at the RR of, say, an ISP in a

[2] For example, users can opt out of the link prefetching mechanism in Chrome.

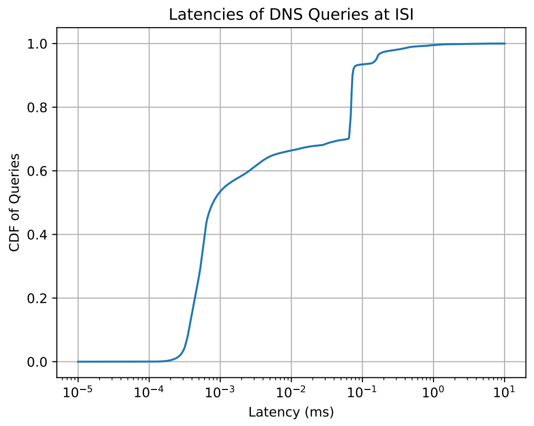

Figure 1: CDF of user DNS query latencies seen at a recursive resolve at ISI, over the course of one year.

large metro area. Nevertheless, we wish to observe some high-level features of the data, and their implications. Figure 1 shows the latencies of all queries seen at ISI over one year. We can see that the latencies are divided into (roughly) 3 regions: sub-millisecond latency, low latency, and high latency. The first region corresponds to cached queries. The second region probably corresponds to DNS resolutions for which the resolving server was close. Finally, the third region probably corresponds to queries that had to travel to distance servers, or required a few rounds of recursion to fully resolve the domain. The results, strikingly similar to those presented in [12], suggest that ISI is no more or less "connected" than the typical recursive resolver.

3.2 How Often Does a User Interact with the Roots?

From packet traces of all of 2018, we would like to quantify the effect caching root DNS records has on users, and how this differs from the ideal behavior users can experience. Specifically, we are interested in 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 cache miss rate, as it approximates how often a TLD record is not found in the cache of the RR 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.

Relevant statistics and their implications on user-perceived latency are presented in Table 1. We find that daily cache miss rates of the resolver range from .1% to 4.5%, with a median value of .5%. This is quite a wide range of cache miss rates, and is potentially skewed by the traffic generated by the internet measurement lab (ISI). However, assessing the latency implications for cache miss rates higher than the median is simply a multiplicative factor and the qualitative conclusions are the same.

Assumptions	Web Page Load Time (ms)	3,000
	Root DNS Latency (ms)	500
	Number of DNS Look-Ups Per Web Page	3
Statistics	Number of User Queries (millions)	14.9
	Number of Root Transactions	73,200
Implications	Percent of user Queries Resulting in a Root Transaction	.49
	Expected Speed-up in PLT with No Root Latency (ms)	8
	Resulting PLT Speedup (percent)	.25

Table 1: Statistics gathered from the ISI RR for a representative month of 2018, and implications of these statistics assuming conservative values for web page load time (PLT), root DNS latency, and number of DNS lookups per page.

To calculate the impact of root latency on user experience, assume that a web page load takes N seconds, that there are M serial DNS requests per page, that the root latency is k seconds, and that the cache miss rate is p. Then, the resulting (average) latency due to root DNS resolution per page load is given by pMk, which, as a fraction of PLT is $\frac{pMk}{N}$.

Although N and k depend on a number of factors difficult to measure, p can be measured for each recursive resolver (as shown in Table 1), and M is relatively constant. To demonstrate this last point, we performed page loads for 500 popular domains using the Selenium web-browser and calculated the number of blocking DNS requests per page load using methods in [33]. We find 95% of requests resulted in $M \leq 3$ and so use this in what follows. As exaggerated upper bounds on the impact of root DNS latency, we choose $N = 3$ and $k = .5$ [3]. Hence, supposing root latency is completely removed from the equation, a user saves, on average, 8 ms per page load. As a percentage of the total page load time, this is .25%, which is measurable, but not significant. Note that for a conservative estimate of the cache miss rate of 4.5%, this would

[3] According to [http archive], RIPE Atlas, and the results from ISI itself, these are quite conservative estimates.

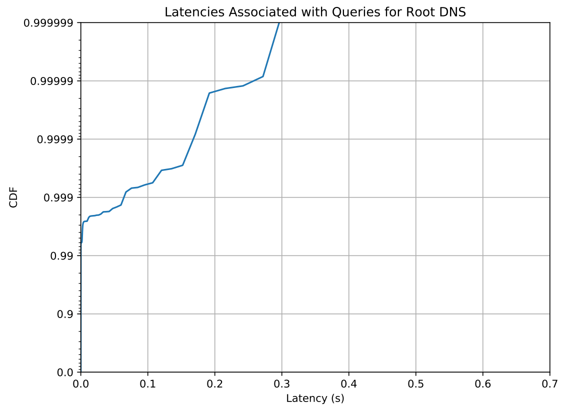

Figure 2: Root DNS latency for queries made by users of the ISI recursive resolver during 2018. user queries that did not generate a query to a root server were given a latency of 0.

translate to approximately 2.3% of a single page load.

As a visualization of these results, Figure 2 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.

Clearly the impact of root DNS latency in this situation is minimal, and we can reasonably conclude that members of the ISI community do not experience much latency due to the root DNS servers. Specifically, Figure 2 suggests it is particularly rare for a user query to generate a query to a root DNS server. Further, Table 1 demonstrates that the latency implications are small for the holistic user population. However, the impact on user performance is 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, this query frequency is absurdly large. This suggests that heuristic arguments that users rarely experience root latency because cached TLD records have long TTL's are not sufficient.

3.3 Case Study - Pointless Root Queries in BIND 9

TODO: Revise Manual inspection of specific query chains in the packet traces suggests that certain logical flows in BIND can result in the root server being unnecessarily queried. We are not making claims that BIND has pathological bugs, since we did not explore the issue further. However, this is an interesting area for future work.

4 A GLOBAL LOOK AT ROOT DNS LATENCY

Looking at cache hit rates of individual RR's is informative, yet does not provide us with a notion of how much latency large user populations experience in a day due to root DNS resolution – a statistic that would help us contextualize the effect of anycast path inflation on user experience. Indeed it would be ideal if we could measure the number of active users, how many times those users request web pages and the cache hit rates of the users' recursive resolvers to obtain an estimate of the latency they experience due to root DNS resolution.

4.1 Data Sources and Processing

Motivated by this ideal scenario, we obtain statistics from a large CDN operator containing information about the users of that CDN. Specifically, these statistics include RR IP's, the number of distinct user IP's behind those RR's and the relative query volume those user IP's contributed (as a fraction of the total CDN query volume) for a month in 2019. Working with the notion of a "user" is valuable, as this provides us with an estimate of how much latency each user experiences. Query volume is also of interest, since it is correlated with activity on the web (and therefore likelihood of waiting for a root DNS query). We saw no significant difference in the following analysis when weighting by user counts or query volume, and so we weight by user counts. We do note, however, that since the notion of user we use is an IP address, one user can actually refer to several people behind a NAT.

To leverage this CDN data in our aforementioned goal, we augment it with traces from the 2018 Day in the Life of the Internet (DITL), organized by OARC. DITL (run annually) contains concurrent packet captures from all root servers except for G-root [4] for two days in Spring. From DITL, we extracted the frequency of requests each RR made to each root, the type of record requested, and the domain requested. Out of a total of 512 billion daily requests to all roots, we observed 310 billion daily requests for bogus domain names and 20 billion daily requests for PTR records. Since these requests are not related to user latency in most cases, we exclude them from the analysis.

[4] A small portion of data from the 2018 DITL is missing or anonymized. We do not compensate for this, as the impact on the results is likely small.

Data Set	Statistic	Percent Representation
DITL ∩ CDN	DITL RR's	29.3
	DITL Volume	71.9
	CDN RR's	78.8
	CDN Volume	88.1
DITL ∩ CDN ∩ RIPE	DITL RR's	.14
	DITL Volume	20.7
	CDN RR's	.34
	CDN Volume	54.6

Table 2: Statistics displaying the extent to which the RR's of users in a large CDN represent RR's seen in the 2018 DITL captures. Also shown is the extent to which RR's of RIPE probes represent the 2018 DITL captures.

We then join these data sets by /24, aggregating their respective user IP counts and query volumes [5] – this joined data set is referred to henceforth as the DCDN data set. In the interest of data sanitation, we remove queries from prefixes in the private IP space [6], as these are not valid queries. Queries from these addresses account for approximately 7% of traffic to the roots. We additionally disregard the presence of IPv6 traffic, and note that it comprises about 12% of DITL query volume. Finally, as previously mentioned, we do not consider requests for bogus domain names nor requests for PTR records. Naturally, there is a mismatch in the /24's represented by each data set. Table 2 summarizes the extent to which the DCDN data (which is a subset of all users in the world) represents the DITL captures, and vice versa. For brevity, we henceforth refer to these /24's as RR's, even though each data point may represent several RR's.

Although the DCDN data considers a relatively small percentage of all RR's, it captures a disproportionately large amount of all DITL volume. A useful visualization of the effect of removing these RR's from consideration is shown in Figure 3. Figure 3 is a CDF of daily query volumes of RR's in the various data sets. The quartiles of the DCDN volume counts (blue) are approximately 10-times those of DITL (orange), suggesting /24s in the DCDN are a particularly high-querying subset of all RR's. Therefore, as we will demonstrate in the following, despite disregarding many RR's, our analysis actually provides an upper bound on root DNS latency experienced by the typical user.

[5]We take care to ensure that neither user counts nor query volumes are "double counted" across different IP's.

[6] The list we use can be found at [36].

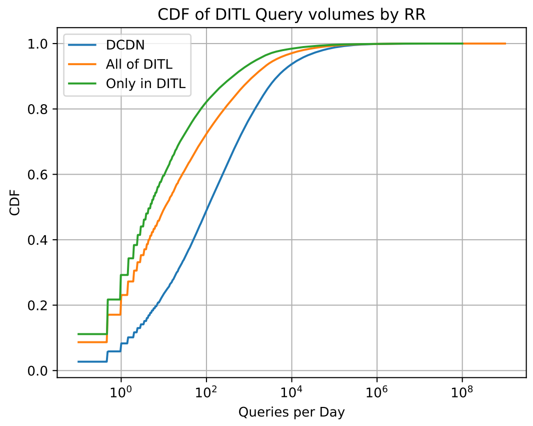

Figure 3: A comparison of daily query volumes seen from various sets of RR's. Those RR's both in the data set of the large CDN and DITL have relatively high daily query volumes, as evidenced by the large tail of DCDN.

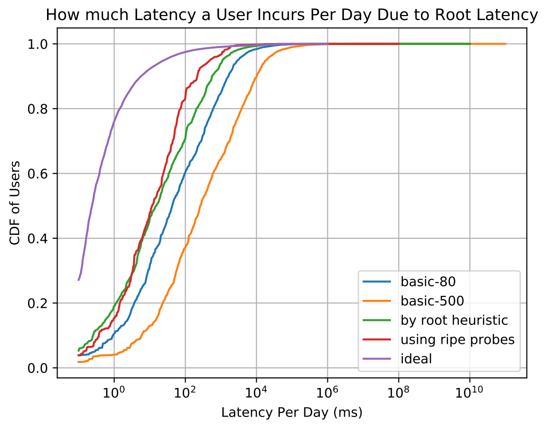

Figure 4: A CDF of approximate latency a user experiences due to root DNS resolution, per day.

4.2 Root Latency Experienced by Users

Our main result, shown in Figure 4, is a CDF of expected user latency per day, where the expected value is calculated according to certain assumptions we make about root latency.

[g]

To generate each line in Figure 4, the expected root latency per RR per query is multiplied by the number of queries per day each RR makes and divided by the number of users that RR represents. The number of queries per day each RR makes is calculated from DITL by first calculating daily query rates at each replica (i.e. total queries divided by total capture time) and subsequently summing these rates across replicas. This product (representing daily query latency per user) is then weighted by user count (from the CDN data set) and the resulting CDF is calculated. Note the only detail still unexplained is the way in which to estimate the expected root latency per RR.

Since we do not know the actual latencies recursives saw to each replica of each root, we use a few methods of estimating this expected latency, each of which has its merits. As a useful comparison, we include lines labeled "basic-**N**" which are generated naively assuming the latency from every RR to every root server is **N** ms. For example, the median latency incurred by a single user each day due to root DNS resolution is 44 ms assuming a query from any RR to each root is 80 ms.

The line labeled "heuristic root value" is calculated as follows: assume each RR queries the set of roots according to a distribution p_{RR}, where p_{RR} is a discrete p.d.f. over the 12 root servers indicating the probability with which the RR chooses each root server to query. This p.d.f. is calculated empirically from the DITL data. Next, obtain median latency values to each root (e.g. from RIPE Atlas) and call these l_{RR}. Note that, for now, l_{RR} is independent of RR. Compute the expected latency from each RR to any root as the dot product between l_{RR} and p_{RR} (i.e. the expectation of the latency). Although somewhat imprecise, this method of calculating latency for each user takes into account that some root servers generally exhibit lower latencies than others due to investments in infrastructure [7]. Hence, this line can be considered a more reasonable estimate of the latency users experience due to root DNS resolution. This is corroborated by the "middle-of-the-pack" median estimate of 15 ms/day.

The line labeled "RIPE" looks only at those RR's in both the DITL and CDN data sets housing RIPE probes. We are interested in RIPE probes, since RIPE probes routinely issue ping measurements to each root server and report latency values. Note that we were able to determine the RR for a subset of all active RIPE probes. Further, the set of RR's housing a RIPE probe did not overlap perfectly with RR's in the DCDN data set. Using our notation from the previous paragraphs, l_{RR} is then populated from these ping measurements for each RR, and the expected root latency per query is calculated per RR according to p_{RR}. From Table 2, we see that /24's housing RR's of RIPE probes only account for 21% of the volume in DITL, and about .1% of all /24's; hence, RIPE probes are not representative of the population (as expected). However, for users in these /24's, this is a fairly accurate measurement of the expected daily latency due to root DNS resolution. Additionally, many studies, e.g. [25], use RIPE probes when measuring anycast latency/inefficiency so this line could provide a useful comparison. We see this method provides one of the lowest median estimates of 12 ms/day. This makes sense, as RIPE probes generally reside in well connected areas of Europe.

Finally, the line labeled 'ideal' does not use DITL query volumes to calculate daily user latency, but instead represents some hypothetical scenario in which each RR queries for all TLD records exactly once per TTL. The resulting hypothetical median daily latency of .23 m[h]s could represent a future in which caching works at recursives as intended.

5 ANYCAST PATH INFLATION

Anycast path inflation is a concept introduced in [16] and expanded upon in [25], which measures how inefficiently queries to anycasted IP's are mapped to physical replicas. We use the definition from [25], which breaks down path inflation into unicast path inflation (UPI) and anycast path inflation (API). UPI measures the extra latency a query incurs as a result of not being mapped to the geographically closest physical replica. API measures the additional latency incurred beyond the optimal unicast alternative. API for a query to an anycasted address can be large, which (generally) occurs when queries are routed to geographically distant replicas, despite the existence of a close replica. We have demonstrated that, due to caching, users do not experience much root latency. Since prior work has focused on inflation (and how it affects users), we further wish to estimate API incurred by users of recursives around the world due to the root DNS servers. Furthermore, we wish to compare this API to inflation incurred by users of a large anycast CDN, to provide perspective as to how this inflation impacts users.

5.1 Root DNS API

To measure anycast inflation for the root DNS deployment, we again leverage the DITL captures. The DITL

[7] For example, at the time of writing the median latency to B root (3 sites) is 130ms whereas the median latency to F root (225 sites) is less than 5 ms.

captures are a rich source of data for this purpose because they provide us with a global view of which recursives are accessing which locations for all but a small subset of root DNS sites [8].

6 DISCUSSION

6.1 Approximations and Limitations

In the above we have attempted to provide upper bounds of the impact of user latency on user experience. The reason for this is two-fold. First, what we are trying to measure can only be measured by accumulating measurements from a variety of sources and so naturally we have a lack of representativeness and certainty when discussing results. Second, we are arguing that users rarely interact with the DNS infrastructure. Hence, we would like to obtain upper bounds on latency experienced by users due to root DNS resolution, so we can make claims with more certainty.

The notable limitations in our above analysis are

(1) In the case of studying root record cache hit rates at the resolver level, we were only able to study one recursive resolver.
(2) When looking at global root server querying behavior, we were only able to calculate daily user latency due to root DNS resolution for approximately 30

However, even when we use extremely conservative estimates of parameters such as page load time, or root DNS latency we find in Section 3.2 that root DNS resolution adds a *negligible* latency to page loads. One might also consider a lack of insight into what latencies each recursive resolver saw to be a weakness of our analysis. However, regardless of which method is used to calculate expected daily user latency, users generally experience median values of < 50 ms/day while worst case users generally experience < 10 s/day. Moreover, recall the following simplifications we make in our above (global) analysis.

(1) Requests to root servers are generated by a myriad of services, so each request may not have been generated by an actual user.
(2) The notion of "user" here (in the CDN) is likely a home router, potentially representing several users across many devices.
(3) The DCDN data set contains particularly "active" RR's.

[8]Notably excluded from this analysis is H root, which did not provide packet traces at the per-site level.

To investigate item 2 in the above list, we looked specifically at recursive resolvers in AS's owned by mobile providers in the United States. Many large mobile providers in the U.S. do not assign IPv4 addresses to individual devices, and so act as giant NAT's. Additionally, U.S. mobile providers have separate AS's for their mobile networks. For this, we use the free IP to ASN mapping service provided by MaxMind. We find that the daily user latency of these particular recursives is indeed orders of magnitude over the median. Since we have no reason to believe mobile Internet traffic generates more root requests than any other type of traffic, this confirms our suspicions that user count estimates for NAT'ed networks are lower bounds. This further establishes that Figure 4 is to be interpreted as a conservative upper bound of daily root latency experienced by users.

6.2 Root DNS in Anycast

In the context of anycast, recall that studies routinely use the root servers as sources of data. However, in light of these results, one can question whether the typical user interacts with these particular anycast deployments. One could go further to argue the utility of proposed improvements to IP anycast should not be assessed using root DNS servers, as the generalization ability of the improvement to all IP anycast systems is uncertain. Regardless, we believe these results should encourage others to consider the implications of anycasts' weakness, whatever they may be. Pointing out sub-optimal performance for the sake of analysis is interesting, but it is equally interesting to note whether or not this sub-optimal performance has any practical, measurable effect.

7 RELATED WORK

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

7.1 Root DNS Anycast Performance

The performance of anycast in the context of root DNS is generally gauged by anycast's ability to balance load among server replicas or provide low latency to users. Generally, all studies conclude that anycast successfully balances load, while latency performance depends on the specific deployment configuration. [30] looks at a DDoS attack on the root name server infrastructure, and generally shows that anycast is a good defense mechanism against such attacks. An earlier study, [32] confirms that anycast protects the root DNS infrastructure against such attacks and, furthermore, that anycast routes users to an optimal location in most cases. [16] looks at user latency to C, F, K, and L-root and attributes better performance to good geographic location and peering strategies. These findings coincide with an earlier study, [10], who conclude the performance of anycast is intrinsically linked to deployment strategy. Additionally [16] finds that as few as 12 sites can provide “good” latency to users. [25], [15], [16], and [26] are all examples of studies who quantify latencies to various root servers, and note how these compare to the (optimal) latency of the closest unicast alternative for the user who issued the query.

7.2 CDN Anycast Performance

Some CDN’s (e.g. Cloudflare, Edgecast, Fastly) use IP anycast to augment their serving infrastructure. When deploying an Anycast CDN (ACDN), delivering content to users with low latency becomes a high priority, as there is a large financial incentive to do so. The simplicity of IP anycast comes at the cost of having coarse grained control over where user queries land. Shifting user load between nodes during peak hours, for example, is a challenging problem. As a potential solution, [6] and [18] use DNS redirects at ADNS servers to shift load among anycast nodes, albeit in slightly different ways. [11] analyzes what latency users are achieving, compared to optimal, when being routed to anycast nodes and finds that 10% of users experience a latency inflation of at least 100 ms.

7.3 Recursive Resolvers and the Benefits of Caching

Similar to the RR analysis conducted here, [21] looks at DNS traffic on a small network and notably finds that 16% of queries resulted in queries to the root, most of which were for invalid domains. As this study is quite old, it is no surprise that this rate has decreased (recall we observed .5% of queries resulted in queries to the root) since browser designers and network engineers understand the importance of caching. [12] also looks at a RR and analyzes statistics of DNS exchanges occurring over it including DNS transaction latencies. Both [35] and [24] look at certain pathological behaviors of popular recursive resolvers, and the implications these behaviors have on root DNS load.

7.4 Web Performance

Although we were unable to find any specific study that looked at how web performance and root DNS latency were related, there are certainly studies characterizing web performance. [33] characterizes web performance bottlenecks in (at the time) new broadband networks, and finds that latency is the main bottleneck for PLT when the user’s bandwidth exceeds 16 Mbps. However, the study does not realistically emulate a page load and, in particular, can not analyze the effect of having multiple DNS resolutions per page. Similarly, [9] analyzes how each step of a page load contributes to the aggregate PLT using a tool designed in-house. However, unlike [33], they did not conduct a large measurement campaign and do not include information about multiple DNS lookups per page. A more recent study, [17] provides a brief survey of web performance measurement studies and explains why it is difficult (with current practices) to compare two different studies in web performance.

8 CONCLUSION

IP anycast has come under attack, with studies showing, for example, how BGP can naturally route users to suboptimal anycast instances and inflate user latencies. Due to the relative availability of root DNS data and diverse deployment strategies of the root DNS servers, they are common targets for delineating inefficiencies and suggesting improvements to IP anycast. We argue not only that the root DNS servers have different design goals (i.e. resiliency against attacks) than that of other anycast services, but also that users rarely interact with the root DNS infrastructure – rendering perceived inefficiencies and proposed improvements to be ill-founded when only tested on the root DNS. Perhaps simple yet effective ideas such as browser link prefetching or DNS request parallelization should be expanded and their adoption by users encouraged, rather than proposed improvements to IP anycast. [a]ideally we would control the linebreak in the latex so it occurs after “it,” [b]this argument about prior studies needs to be more careful, I think. I propose an alternate abstract below [c]perhaps add others [d]does markdown not handle blank-lines as paragraph breaks? this

stuff is a step backwards. [e]we have to be careful about this statement. “has its drawbacks” is really vague–every study has drawbacks. I think the point is stronger: the IMPLICATION of these studies is performance is much worse than it could be and people are suffering. We show that (1) no, it doesn’t matter for the root.

But we need to go further than that. Does the root STAND IN for other anycast systems? Even if the root isn't slow in practice, would cloudflare be slow? (2) we need to say how we think our results should generalize (or that they only apply to the root) [f]again, "contextualize resarch" : who cares. Our goal is (I think) to show that concerns raised about the root latency are greatly exaggerated. ALSO we need to pick out a specific claim or two, and say what we think to be a more represntive number. That is: we need to find a number like 100ms and show that 2ms is what is perceived. [g]update labels in legend [h]assumed 80 ms per query – should comment on this, or just quote the expected number of queries per day, this might be more intuitive [i]interesting that in 10 minutes they observe so many queries for COM TLD yet don't see any issue with that [j]Might be an interesting tool to use [k]Mark shared in an email – shows time between DNS queries & TCP connection starts can be big, which suggests DNS is not blocking

REFERENCES

[1] root-servers.org

[2] akamai.com/us/en/multimedia/documents/white-paper/akamai-designing-dns-for-availability-and-resilience-against-ddos-attacks.pdf

[3] dns-oarc.net/oarc/data/ditl/2018

[4] The Top 500 Sites on the Web. alexa.com/topsites

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

[6] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van Der Merwe. 2011. A practical architecture for an anycast CDN. *ACM Transactions on the Web (TWEB)* 5, 4 (2011), 17.

[7] Amazon. aws.amazon.com/route53/faqs/

[8] Amazon. aws.amazon.com/cloudfront/

[9] A. S. Asrese, P. Sarolahti, M. Boye, and J. Ott. WePR: a tool for automated web performance measurement. In *2016 IEEE Globecom Workshops (GC Wkshps)*,. IEEE, 1–6 2016.

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

[11] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye. Analyzing the Performance of an Anycast CDN. In *Proceedings of the 2015 Internet Measurement Conference*,. ACM, 531–537 2015.

[12] T. Callahan, M. Allman, and M. Rabinovich. 2013. On modern DNS behavior and properties. *ACM SIGCOMM Computer Communication Review* 43, 3 (2013), 7–15.

[13] Cloudflare. Load Balancing without Load Balancers. blog.cloudflare.com/cloudflares-architecture-eliminating-single-p/

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

[15] L. Colitti, E. Romijn, H. Uijterwaal, and A. Robachevsky. 2006. Evaluating the effects of anycast on DNS root name servers. *RIPE document RIPE-393* 6 (2006).

[16] R. de Oliveira Schmidt, J. Heidemann, and J. H. Kuipers. Anycast Latency: How Many Sites Are Enough?. In *International Conference on Passive and Active Network Measurement*,. Springer, 188–200 2017.

[17] T. Enghardt, T. Zinner, and A. Feldmann. Web performance pitfalls. In *International Conference on Passive and Active Network Measurement*,. Springer, 286–303 2019.

[18] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. Fastroute: A scalable load-aware anycast routing architecture for modern CDNs. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*,. 381–394 2015.

[19] S. Goldlust. Early Refresh of Cache Records. kb.isc.org/docs/aa-01122

[20] Google. developers.google.com/speed/public-dns

[21] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on networking* 10, 5 (2002), 589–603.

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

[23] J. Kurose and K. Ross. 2010. Computer networks: A top down approach featuring the internet. *Peorsoim Addison Wesley* (2010).

[24] M. Lentz, D. Levin, J. Castonguay, N. Spring, and B. Bhattacharjee. D-mystifying the D-root Address Change. In *Proceedings of the 2013 conference on Internet measurement conference*,. ACM, 57–62 2013.

[25] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee. Internet anycast: performance, problems, & potential.. In *SIGCOMM*,. 59–73 2018.

[26] J. Liang, J. Jiang, H. Duan, K. Li, and J. Wu. Measuring query latency of top level DNS servers. In *International Conference on Passive and Active Network Measurement*,. Springer, 145–154 2013.

[27] S. McQuistin, S. P. Uppu, and M. Flores. Taming Anycast in the Wild Internet. In *Proceedings of the Internet Measurement Conference*,. 165–178 2019.

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

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

[30] G. Moura, R. d. O. Schmidt, J. Heidemann, W. B. de Vries, M. Muller, L. Wei, and C. Hesselman. Anycast vs. DDoS: Evaluating the November 2015 root DNS event. In *Proceedings of the 2016 Internet Measurement Conference*,. ACM, 255–270 2016.

[31] C. Partridge, T. Mendez, and W. Milliken. Host Anycasting Service. tools.ietf.org/html/rfc1546

[32] S. Sarat, V. Pappas, and A. Terzis. On the use of anycast in DNS. In *Proceedings of 15th International Conference on Computer Communications and Networks*,. IEEE, 71–78 2006.

[33] S. Sundaresan, N. Magharei, N. Feamster, R. Teixeira, and S. Crawford. 2013. Web performance bottlenecks in broadband access networks. *ACM SIGMETRICS Performance Evaluation Review* 41, 1 (2013), 383–384.

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

[35] Y. Yu, D. Wessels, M. Larson, and L. Zhang. 2012. Authority server selection in DNS caching resolvers. *ACM SIGCOMM Computer Communication Review* 42, 2 (2012), 80–86.

[36] zakird. Private IP List. github.com/zmap/zmap/blob/master/conf/blacklist.conf