Impact of Root DNS Latency on User Experience

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

ABSTRACT

Anycast is means of distributing content that has been praised for its simplicity and performance, yet criticized for inflating client latencies in some cases. The root DNS servers are frequent sources of information for studies analyzing anycast, since the information is relatively easy to obtain. We argue that the root DNS servers are not valid test subjects for studies either offering criticism of or suggesting improvements for anycast, since clients rarely interact with the root DNS infrastructure. We demonstrate that simple caching policies of recursive resolvers limit a clients' exposure to the root DNS infrastructure, and quantify how much latency a client experiences each day due to root DNS resolution. These results indicate that future studies should not draw from root DNS server data to support arguments regarding anycast latency inflation.

1 INTRODUCTION

Anycast is a method of server deployment in which a set of physically distinct servers, called anycast replicas, all serve the same content. Specifically, IP anycast is a system in which geographically diverse anycast replicas all advertise the same IP address using the Border Gateway Protocol (BGP). Routing policies of providers and BGP determine which anycast replica a client will be mapped to when querying the (anycasted) IP. This is all transparent to the client, as each site has exactly the same content [23], [15].

Anycast has been lauded for its potential to offer improved latency and decreased load on each anycast server, doing so with minimal scaling complexity. The geographical diversity of anycast deployments offers resilience to network outages and targeted attacks [18], [22]. However, numerous studies have argued that anycast can lead to reduced performance for a subset of clients. Some go further to say that adding more sites may harm, rather than help user performance [18], [24]. The basic idea behind these claims is that BGP can sometimes direct clients to far-away anycast servers, despite the existence of a geographically close server. This inefficiency unnecessarily inflates latencies for users, and harms their experience.

The root DNS servers notoriously feature in studies involving anycast because it is relatively easy to gain access to root DNS data, information about their deployments, and the fact that 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 [18].

Using the root DNS servers to draw conclusions about the performance of anycast as a whole has its drawbacks. Organizations in charge of managing these services have different performance goals when deploying new servers. 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 [22]. Conversely, 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. This is on top of the infrastructure the CDN may already have in place, such as a diverse set of peers or a private wide area network (WAN). CDN's are very attentive of user-perceived latencies, as higher latency has been directly shown to affect profits [20]. 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.

Nevertheless, in the face of perceived anycast inefficiencies many authors have offered solutions ranging from ISP policy changes to modifications to BGP to bolster the performance of anycast [18], [26]. Often, studies have framed the problem as a sort of optimization problem, where every client request should be mapped to the same physical site as the best unicast alternative. These proposed modifications are then

shown to push anycast closer to optimal performance. We argue that, in the context of root DNS query resolution, such modifications and improvements would add negligible benefits to end-users. Specifically, we aim to show that the typical user almost never directly queries the root DNS server. We further place an upper bound on the amount of latency a typical user of a large CDN experiences due to root DNS resolution each day, and show that this latency is negligible. In light of these results, in the context of root DNS anycast deployments, any proposed improvements for anycast shown to decrease latency for users are of questionable practical utility.

2 DNS – A PRACTICAL VIEWPOINT

Modern day DNS infrastructure is designed so as to limit the time a user has to wait for DNS query resolution, and specifically root DNS query resolution. Herein we describe the implementation of DNS, both client and server side, and discuss how this allows for faster DNS resolution at the client.

2.1 Great Latency Savings Potential

There are many descriptions of DNS [16], [8], and we by no means 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 [21]. 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 [7]. 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 [13]. 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[c][d][e][f][g][h]. One might speculate that, since the most popular websites fall into ten or so popular TLD's [2], that a request to the root server would rarely occur. In this scenario, a user fetching example.com would only generate a request for the COM NS record once every 2 days. Furthermore, some commercial RR's are 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. 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 or a conservative manager of the RR, as we will now discuss.

[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.

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

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 [27]. 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. Users issuing reverse-DNS lookups or requesting invalid domains always result in queries to a root server (although it can be argued that these should not be considered here). 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, we analyze packet traces of a recursive resolver at ISI. The recursive resolver (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. Nevertheless, we wish to observe some high-level features of the data, and their implications. 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 call this metric the cache miss rate, as it approximates how often a TLD record was not found in the cache of the RR in the event of a client query. We say approximately since, for example, the recursive resolver may have sent multiple root requests per client query, or root requests without any client query triggering them.

Assumptions	Web Page Load Time (ms)	1,000
	Root DNS Latency (ms)	500
	Number of DNS Look-Ups Per Web Page	10
Statistics	Number of Client Queries (millions)	14.9
	Number of Root Transactions	73,200
Implications	Percent of Client Queries Resulting in a Root Transaction	.49
	Expected Speed-up in PLT with No Root Latency (ms)	24.5
	Resulting PLT Speedup (percent)	2.45

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.

Relevant statistics and their implications on user-perceived latency are presented in Table 1. We found that daily cache miss rates of the resolver ranged from .1% to 4.5%, with a median value of .5%. This was quite a wide range of cache miss rates, and was 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. To obtain an upper bound on the impact of root latency, assume (conservatively) that a web page load takes 1 second, that there are 10 serial DNS requests per page, and that the root latency is 500ms [3]. The numbers used here are only exaggerated upper bounds on the impact of root DNS latency, and are only meant

[3] According to [http archive], and RIPE Atlas, these are quite conservative estimates.

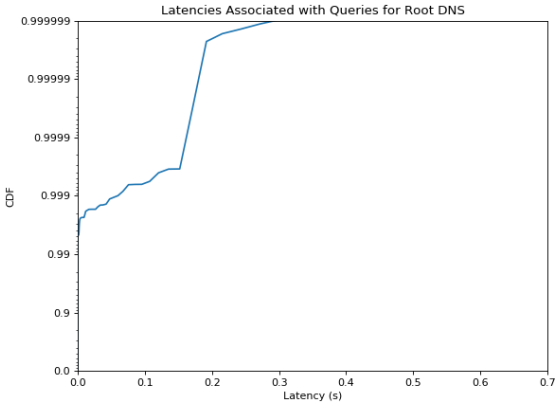

Figure 1: Root DNS latency for queries made by clients of the ISI recursive resolver during 2018. Client queries that did not generate a query to a root server were given a latency of 0.

to provide context. Supposing root latency is completely removed from the equation, a user saves, on average, 27 ms per page load. As a percentage of the total page load time, this is 2.7%, which is measurable, but not significant. Note that for a conservative estimate of the cache miss rate of 4.5%, this would translate to approximately 24.3% of a single page load. To provide a sort of visual context for these results, Figure 1 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 1 suggests it is particularly rare for a client query to generate a query to a root DNS server; further Table 1 demonstrates that the latency implications are small for the holistic client population. However, the impact on user performance is not as small as one would expect. For example, a particular day during January 2018 saw 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. 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 USERS OF A LARGE CDN[I]

Looking at cache hit rates of individual RR's is informative, yet does not provide us with a notion of how much latency each user experiences in a day due to root DNS resolution. Indeed it would be ideal if we could measure the number of active clients, how many times those clients request web pages and the cache hit rates of the clients' 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 client IP's behind those RR's and the relative query volume those client IP's contributed (as a fraction of the total CDN query volume[j]) 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. However, since the notion of 'user' is an IP address, one 'user' can actually refer to several people behind a NAT. Furthermore, when weighting results, client query volume is intuitively the correct metric to weight by, since query volume indicates activity on the web. [k][l]The relationship between daily average query volume and daily average client IP count per RR is shown in Figure 2. The median correlation coefficient between weekly volume and client count is .83, with actual daily correlations ranging from .78 to .83. This correlation is not perfect, but good enough for us to represent the "importance" of an RR by the number of distinct client IP's behind it.

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.

[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.

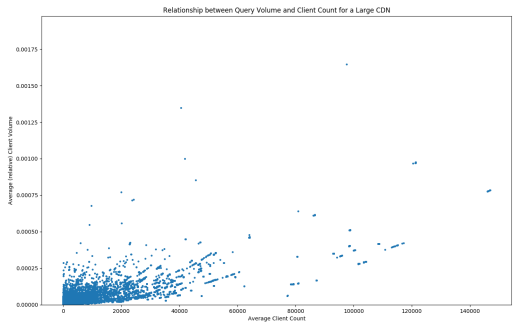

Figure 2: Relationship between daily average client count and daily average (relative) query volume for users of a large CDN. Each point represents one RR.

Data Set	Statistic	Percent Representation
DITL ∩ CDN	DITL RR's	25.7
	DITL Volume	71.4
	CDN RR's	80.9
	CDN Volume	88.4
DITL ∩ CDN ∩ RIPE	DITL RR's	.14
	DITL Volume	20.9
	CDN RR's	.4
	CDN Volume	55.4

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.

Since these requests are not related to user latency in most cases, we exclude them from the analysis.

We then join these data sets by /24, aggregating their respective client IP counts, query volumes, etc... – this joined data set is referred to henceforth as the DCDN data set [29 and 16 in yahoo video]. To determine how "representative" this DCDN data set is, we remove queries from prefixes in the private IP space [5], 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. 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) 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.

[5] The list we use can be found at [28].

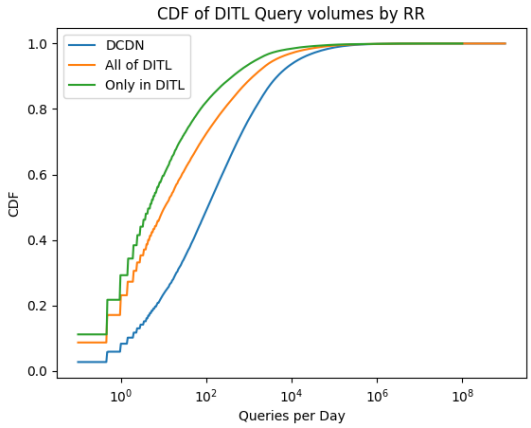

Figure 3: A comparison of daily query volumes seen from various sets of RR's. Those RR's both in the data set of the large CDN and DITL have relatively high daily query volumes, as evidenced by the large tail of DCDN.

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. The quartiles of the DCDN volume counts (blue) are approximately 10-times those of DITL (orange), suggesting /24s in the DCDN are a particularly active subset of all RR's. This suggests that, despite disregarding many RR's, our analysis actually provides an upper bound on root DNS latency experienced by the typical user.

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.

[m]

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. This 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.

We use a few methods of calculating this expected latency. As a useful comparison, we include lines labeled "basic-n" which are generated naively assuming

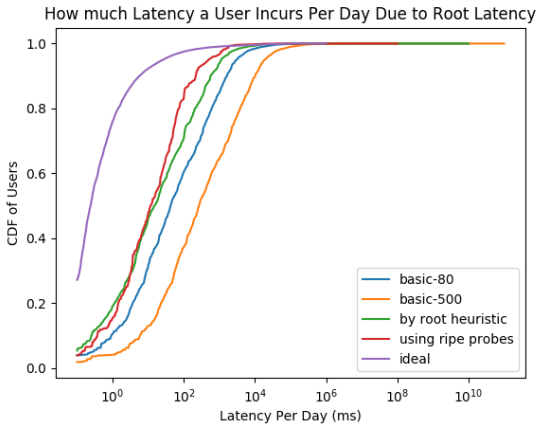

Figure 4: A CDF of approximate latency a user experiences due to root DNS resolution, per day.

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 [6]. 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 clients 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. [18], 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 ms could represent a future in which caching works at recursives as intended.

4.3 Discussion

4.3.1 Approximations and Limitations. Regardless of which method is used to calculate expected 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 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) We do not prune requests unrelated to typical web traffic such as PTR requests or requests to invalid domains.
(4) The DCDN data set contains particularly "active" RR's.

This further establishes that Figure 4 is to be interpreted as a conservative upper bound of daily root latency experienced by users. Although it may not be valid or worthwhile to reason about specific numbers obtained from the above

[6] 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.

analysis, we can say that the typical user experiences very little latency due to root DNS resolution.

4.3.2 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.

5 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 client-centric measurements of web performance.

5.1 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. [22] 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, [24] confirms that anycast protects the root DNS infrastructure against such attacks and, furthermore, that anycast routes users to an optimal location in most cases. [10] 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, [5], who conclude the performance of anycast is intrinsically linked to deployment strategy. Additionally [10] finds that as few as 12 sites can provide "good" latency to users. [18], [9], [10], and [19] 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 client who issued the query.

5.2 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 client queries land. Shifting client load between nodes during peak hours, for example, is a challenging problem. As a potential solution, [12] and [3] use DNS redirects at ADNS servers to shift load among anycast nodes, albeit in slightly different ways. [6] analyzes what latency clients are achieving, compared to optimal, when being routed to anycast nodes and finds that 10% of clients experience a latency inflation of at least 100 ms.

5.3 Recursive Resolvers and the Benefits of Caching

Similar to the RR analysis conducted here, [14] 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. [7] also looks at a RR and analyzes statistics of DNS exchanges occurring over it including DNS transaction latencies. Both [17] and [27] look at certain pathological behaviors of popular recursive resolvers, and the implications these behaviors have on root DNS load.

5.4 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. [25] 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, [4] analyzes how each step of a page load contributes to the aggregate PLT using a tool designed in-house. However, unlike [25], they did not conduct a large measurement campaign and do not include information about multiple DNS lookups per page. A more recent study, [11] 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.

6 CONCLUSION

IP anycast has come under attack, with studies showing, for example, how BGP can naturally route clients to suboptimal anycast instances and inflate client 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]i'm not sure where this author order came from, but i'm pretty certain I shouldn't be first author. Tom? [b]I haven't put these in any particular order yet [c]maybe generate a figure showing a CDF of TTL's of all TLD NS records (pretty easy to generate) [d]Would probably need to run BIND locally, and set myself as my own recursive resolver [e]+ethanbkb@gmail.com

What do you think? [f]*Marked as resolved* [g]*Re-opened* Seems reasonable [h]I did this and all but 2 of the 1000+ had a TTL of 2 days. The other 2 had 1 day. [i]TODO: redo this section in light of Private IP removal, Squat Space removal, removal of PTR requests, removal of invalid TLD's [j]perhaps elaborate on what is meant by "query volume" [k]+ethanbkb@gmail.com This is one way I currently use Microsoft's query volume numbers to justify our results. What do you think? [l]I was thinking along the lines of weighting root query cost by query volume (rather than by "users"). [m]update labels in legend [n]interesting that in 10 minutes they observe so many queries for COM TLD yet don't see any issue with that [o]Might be an interesting tool to use [p]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] The Top 500 Sites on the Web. https://www.alexa.com/topsites

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] Cloudflare. What is DNS? https://www.cloudflare.com/learning/dns/what-is-dns/

[9] 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).

[10] R. de Oliveira Schmidt, J. Heidemann, and J. H. Kuipers. ?. In *International Conference on Passive and Active Network Measurement*,. Springer, 188–200 2017.

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

[12] 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.

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

[14] 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.

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

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

[17] 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.

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

[19] 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.

[20] G. Linden. Make Data Useful. https://sites.google.com/site/glinden/Home/

[21] P. Mockapetris. Domain Names - Implementation and Specification. https://www.ietf.org/rfc/rfc1035.txt

[22] 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.

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

[24] 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.

[25] 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.

[26] J. Xue, W. Dang, H. Wang, J. Wang, and H. Wang. Evaluating performance and inefficient routing of an anycast CDN. In *Proceedings of the International Symposium on Quality of Service*,. ACM, 14 2019.

[27] 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.

[28] zakird. Private IP List. https://github.com/zmap/zmap/blob/master/conf/blacklist.conf