On Eliminating Root Nameservers from the DNS

Mark Allman
International Computer Science Institute

ABSTRACT

The Domain Name System (DNS) leverages nearly 1K distributed servers to provide information about the root of the Internet's namespace. The large size and broad distribution of the root nameserver infrastructure has a number of benefits, including providing robustness, low delays to topologically close root servers and a way to cope with the immense torrent of queries destined for the root nameservers. While the root nameserver service operates well, it represents a large community investment. Due to this large cost, in this paper we take the position that DNS' root nameservers should be eliminated. Instead, recursive resolvers should use a local copy of the root zone file instead of consulting root nameservers. This paper considers the pros and cons of this alternate approach.

ACM Reference Format:
Mark Allman. 2019. On Eliminating Root Nameservers from the DNS. In *Proceedings of The 18th ACM Workshop on Hot Topics in Networks (HotNets'19).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3365609.3365863

1 INTRODUCTION

Most network transactions begin with one or more queries of the Domain Name System (DNS) [24]—making DNS a foundational component of the Internet. All names within the DNS branch off in a tree-like fashion from a common root. The task of answering queries for the root of the namespace is assigned to 13 named authoritative servers. Each of these nameservers is replicated—using anycast routing—such that in total nearly 1K instances of the root nameservers are currently in operation. This level of replication has several benefits, including (*i*) providing a robust service that can easily cope with individual servers being temporarily offline, (*ii*) having a server close to many Internet users and hence providing for low delays when querying the roots and (*iii*) being able to share and handle the immense torrent of queries that arrive at the root nameservers. Previous studies of traffic arriving at the root nameservers show that more than 95% of the queries are "junk" of some form—ranging from requests for non-existent record types or top-level domains (TLDs) to spuriously repeated queries [8, 9, 23, 31]. This basic finding was first reported in 2001 [8], but has held each time researchers have re-appraised the initial work. In §2.2 we illustrate that this basic result still holds with current data. Given that we use a significant and expanding infrastructure (see §2.1) to support the root namespace and, yet, most of the effort is spent on bogus requests, a natural question is:

> *Is a service where more than 95% of the effort is fruitless correctly architected, or is there a better structure?*

On the one hand, this question is nonsensical. A public service like the DNS root nameservers cannot control the kinds or amount of traffic that arrives. Further, properties such as robustness, correctness, security, agility and timeliness are more important goals than designing to limit wasted effort. Therefore, the "junk" traffic can be viewed as simply one of the costs we must cope with to operate an effective system.

However, in this paper, we take a contrary position: While we cannot eliminate the root of the namespace, we can and should eliminate the root nameservers. Instead, recursive resolvers should simply use a local copy of the root zone file rather than querying a root nameserver for the information. We argue that the costs of the enormous infrastructure the community has put in place to ensure the DNS roots operate as expected are not only quite high, but the benefits compared to simply distributing the root zone file are low.

2 THE DNS ECOSYSTEM

Before discussing our proposal (§3), we highlight several facets of the DNS ecosystem to provide background.

2.1 Configuration

We first sketch several mostly static aspects of how the DNS ecosystem is configured.

Root Hints File: Currently end-user devices generally send all DNS queries to a recursive resolver which is configured via DHCP when joining a network. Alternatively, a user may configure their device to leverage a public recursive resolver (e.g., Google's 8.8.8.8 resolver). The recursive resolver queries authoritative nameservers—which hold the actual

HotNets'19, November 14-15, 2019, Princeton NJ, USA

ACM ISBN 978-1-4503-7020-2/19/11…$15.00
https://doi.org/10.1145/3365609.3365863

name-to-address bindings—on behalf of the client. Recursive resolvers use the so-called "root hints" file to bootstrap the process of finding authoritative nameservers. The root hints file contains a list of all 13 named authoritative root nameservers, which are denoted *a*-root–*m*-root. For each named root, the file includes the nameserver (NS record), as well as an IPv4 address (A record) and an IPv6 address (AAAA record) for the nameserver. Therefore, the root hints file has 39 total entries and is roughly 3KB. With this information, the recursive resolvers can start the lookup process at the root and proceed as directed in DNS replies. For instance, a reply from a root nameserver will direct the resolver to contact a specified TLD nameserver. Each record in the root hints file has a TTL of 3.6M seconds—or roughly 42 days. The file is largely static, but does change in some cases and a fresh copy should be obtained by the recursive resolvers upon expiration. However, there is evidence that suggests a significant number of recursive resolvers do not update their root hints file in a timely fashion [23].

Root Zone File: Each root nameserver answers queries based on a copy of the root zone file—which is authoritatively provided by ICANN and distributed by Verisign. The root nameservers do not directly answer queries, but refer the requester to the nameserver for the TLD of the hostname in the query. For instance, a request for "www.sigcomm.org" will cause the root nameserver to refer to the requester to the authoritative server for ".org, which will in turn refer the requester to the authoritative nameserver for "sigcomm.org". We have an archive of one snapshot per day of the root zone file since April 28, 2009. Figure 1 shows the number of records in the root zone on the 15^{th} of each month. The figure shows that after a period of stability the number of records in the root zone file increased over five-fold between early 2014 and early 2017. This was caused by an increase in the number of TLDs—from 317 TLDs on June 15, 2013 to 1,534 TLDs on June 15, 2017. After this period of growth the size has stabalized to roughly 22K entries. The current root zone file is roughly 1.1 MB compressed. Currently, the TTLs of the NS, A and AAAA records in the root zone file are two days. Therefore, a recursive resolver can cache the responses from the root nameservers and continue to use these in answering queries from end-user devices for two days.

Scope of Root Nameservers: As we note above, there are 13 named authoritative root nameservers. However, each named root nameserver consists of multiple replicas setup via anycast routing. On May 15, 2019, *root-servers.org* reported 985 root nameserver instances. Twelve different organizations run the root nameservers[1] and each applies its own replication strategy. Therefore, the number of replicas per named root varies from at most six instances for *b,g,h,m*-root to over 100 instances for *d,e,f,j,l*-root. Figure 2 shows the total number of instances on the 15^{th} of each month since March 2015, as reported by *root-servers.org*. There are several large jumps in the number of instances caused by *e*-root and *f*-root, as follows: (*i*) between January and February, 2016 *e*-root added 45 instances, (*ii*) between April and May, 2017 *f*-root added 81 instances and (*iii*) between November and December, 2017 *e*-root added 85 instances and *f*-root added 43 instances. In addition to these large jumps, the plot shows the number of root nameserver instances—and their attendant cost—is increasing steadily over time.

[1]Verisign operates both *a*-root and *j*-root.

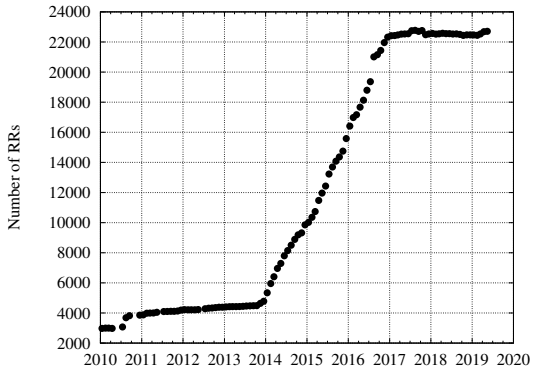

Figure 1: Num. of records in the root zone over time.

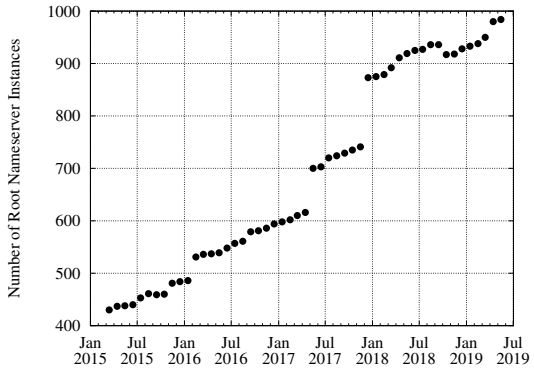

Figure 2: Root nameserver instances over time.

2.2 Root Nameserver Traffic

We now turn from the configuration of the DNS ecosystem to the operation of the root nameservers. Previous work shows more than 95% of the queries arriving at the root nameservers are bogus in some fashion [8, 9, 23, 31]. For instance, [9] studies eight root nameservers and shows that in 2008 only 1.8% of the queries are valid. Here we aim to both (*i*) validate these previous findings still roughly hold and (*ii*) illustrate

the magnitude of queries arriving at the root nameservers. For this we leverage the Day-In-The-Life (DITL) 2018 dataset provided by *j*-root and DNS-OARC [13]. We analyze 24 hours of traffic to 142 instances of *j*-root on April 11, 2018.[2] The instances are geographically spread across the world. Keep in mind that our analysis uses data from only one of the 13 root servers. While this means we analyze only a fraction of the traffic to the root nameservers, previous work shows that the type of traffic arriving is fairly consistent across roots [9]. In total, *j*-root received roughly 5.7B queries in 24 hours—or, about 66K queries/second. The queries were from 4.1M distinct resolvers (IP addresses). However, 723K of the resolvers only query for bogus TLDs—meaning at most only 3.4M resolvers are accomplishing useful work.

As in previous studies, we find that the bulk of the requests are junk. We find requests with bogus TLDs account for 3.5B queries—or, 61.0% of the total. In addition, if we assume ideal caching at the resolvers—so, the resolver should make a single request for each needed TLD within our 24 hour dataset—we find an additional 2.2B queries (38.4%) are invalid. In this case, the queries for bogus TLDs (61.0%) and the requests for records that should have been cached (38.4%) leave just 0.5% of the queries as valid. If we relax the assumption of an ideal cache and instead allow resolvers to make a valid request for each TLD every 15 minutes—or 96/day—the number of spurious requests is 2.0B (35.7%). In this case, the queries for bogus TLDs (61.0%) and requests for records that should have been cached (35.7%) leaves 3.3% of the queries as valid—or just 187M of the 5.7B total queries. This means that each instance of *j*-root is dealing with roughly 15 valid queries/second on average. This shows that the high-order results from previous research still hold: the vast majority of traffic arriving at root nameservers is junk.

3 PROPOSAL: ELIMINATE ROOT NAMESERVERS

Our position in this paper is simple: *eliminate all authoritative root nameservers*. Instead of recursive resolvers bootstrapping with a root hints file that facilitates transactions with root nameservers, the recursive resolvers will directly use their own local copy of the root zone file to find TLD nameservers. Below we comment on several practical aspects of enacting this approach before considering the benefits (§4) and costs (§5) in subsequent sections.

Cryptographically Sign Root Zone: Since July 2010 the contents of the root zone have been cryptographically protected via DNSSEC [3–5]. Therefore, a recursive server obtaining a copy of the root zone file for direct use can validate the integrity of the contents. As an optimization the entire root zone file could be cryptographically signed such that it can be validated quickly rather than validating each component individually.

Root Zone Distribution: A mechanism for providing root zone files will need to replace the traditional root nameservers. Given the integrity of the zone file will be cryptographically secure, the delivery can take many forms and develop organically. For instance, the root zone could be distributed via a set of HTTP mirrors as we use for software distribution. Or, a public recursive server may provide the root zone via DNS' own zone transfer mechanism.[3] Alternatively, the root zone could be shared via BitTorrent or a similar peer-to-peer system. Finally, an rsync [2, 30] server or similar system could be used such that only changes in the root zone file would need to propagate instead of the entire file.

Implementation: After obtaining and verifying the root zone file, recursive resolvers must have a way to incorporate the records into their lookup process. A first option is for the resolver to simply read all records in the root zone and place each in the resolver's local cache, just as if the information come in a DNS response from a root nameserver. This has the advantage of not requiring further per-transaction effort. However, it may pollute the cache with unneeded records. An alternative is to engineer the recursive resolver to consult the local root zone file each time it would currently consult a root nameserver. The needed records could then be cached by the resolver as usual. This has the benefit of not polluting the cache with unused records, but it requires ongoing effort as the recursive encounters TLDs for the first time. Finally, an operator may simply make the root zone file available to its resolvers via an authoritative server accessible only by the internal recursive resolvers, as suggested in RFC 7706 [22].[4] While the downside of this approach is running an extra server, some may find it appealing as the recursive resolvers do not need to change.[5]

Deployment: A final consideration is deployment. Our approach allows each recursive resolver to independently abandon the root nameservers. Therefore, our approach does not require a flag day on which all resolvers must switch. This also means that the root nameserver infrastructure can be gradually rolled back over some period of time as the number of resolvers using root nameservers diminishes. Of course, as the scope of the root nameserver infrastructure shrinks, performance may suffer. However, this drawback may provide an incentive for recursive resolvers to start using their own local copy of the root zone file.

[2] According to *root-servers.org*, *j*-root had 160 replicas on April 11, 2018. The DITL dataset does not indicate why only 142 replicas are included in the dataset, but there is nothing about the missing set that suggests bias.

[3] The root zone is currently available via zone transfer from ICANN [19].

[4] RFC 7706 suggests running a local instance of a root nameserver as an optimization. We discuss the relationship to this work in §6.

[5] Aside from a configuration change to consult the local root nameserver instance instead of those found in the standard root hints file.

4 BENEFITS

We now turn to the benefits of eliminating root nameservers.

Less Infrastructure: One obvious benefit of eliminating root nameservers is simply to remove the non-trivial infrastructure the community now operates and its attendant costs. As we discuss in §2, we currently operate nearly 1K instances of the root nameservers to ensure availability and timeliness. Further, this is not a fixed cost. As Figure 2 shows, the number of root nameserver instances has more than doubled over the last four years and this trend shows no signs of abating. Further, even if the number of root nameserver instances were to level off, the current cost is still significant in terms of both number of machines and coordination. Also, since at least 2001 we know most of the requests processed by the root nameservers are bogus. These bogus requests are at least part of what necessitates an increasing number of root nameserver instances and therefore it is doubtful that the number of instances will level off. Recursive resolvers must already bootstrap using a file that lists starting points. By changing this file from the traditional root hints file to the root zone file the community can decommission a large infrastructure.

Performance: Using a local copy of the root zone file instead of querying a root nameserver can save a network transaction each time a resolver needs to determine the authoritative nameservers for a TLD. This optimization is one of the motivations for RFC 7706 which discusses running a local root nameserver [22]. We expect this optimization to be modest at best since the TTLs for TLDs are two days, making the records highly cacheable. Therefore, recursive resolvers should visit root nameservers infrequently—especially for popular TLDs—which means the performance savings from eliminating root nameservers is also likely to be overall small.

Cache Capacity: Depending on how a recursive resolver incorporates the root zone file, having the information locally may free some in-memory cache space. Consider the case where a resolver can quickly query a local database containing the root zone file. In this case there is little reason to keep the TLD records in the resolver's in-memory cache as they can be cheaply retrieved from the database as needed. This can then free in-memory cache space that is normally used for TLD records for other records. This benefit depends on how the resolver incorporates the root zone file and therefore this is not a guaranteed benefit of using a local copy of the root zone file.

Robustness: Eliminating root nameservers makes the system more robust since resolvers will no longer rely on the root nameservers during the resolution process and therefore that is one less thing that can go wrong and hamper a DNS lookup. While a recursive resolver must still retrieve the root zone file, there is some natural robustness to the process given (*i*) the lengthy TTLs in the root zone and (*ii*) that retrieving the root zone file will be an out-of-band process. For instance, a recursive resolver that obtains the root zone file at time X could attempt to update its copy at time $X + 42$ hours. If the retrieval fails, the resolver has 6 hours to re-try before its current root zone file expires and there is an actual impact on DNS lookups. In reality, the robustness benefits to eliminating the roots will be fairly minor precisely because our investment in a large root nameserver infrastructure means that resolvers generally can cope with failure by leveraging a different root nameserver. That said, even if our approach does not add practical robustness it achieves robustness at a much lower cost.

Security: While recent proposals call for DNS to run over TCP+TLS [12, 18, 32] or HTTPS [17], DNS' operation is still mostly conducted in cleartext over UDP.[6] This means DNS is susceptible content-based attacks—e.g., man-in-the-middle attacks [10, 27], cache poisoning attacks [16, 21, 27]. Therefore, by eliminating some DNS transactions from traversing the network we reduce the attack surface. Further, [20] argues that so-called "root manipulation" is a particularly nasty form of man-in-the-middle attack. First, it is relatively easy for a nefarious network operator—e.g., in the name of censorship—to identify queries to root nameservers since they will all be destined for one of 13 IP addresses. Therefore, it is also relatively easy to provide fraudulent responses by either answering these queries directly as they are observed or by diverting them to nameservers masquerading as root nameservers. Second, since all hostnames share the root of the namespace, hijacking root queries can give an attacker control of the entire namespace. While [20] finds only small amounts of root manipulation, the threat remains. Eliminating the root nameservers does not eliminate all threats posed by cleartext DNS transactions, but it does remove one significant angle of attack.

Privacy: Similar to security issues, DNS has privacy issues in its usual operational mode of cleartext transactions over UDP. This allows monitoring at any point along the path a transaction travels. Running DNS over TCP+TLS [12, 18, 32] or HTTPS [17] thwarts such monitoring, but is not yet common practice. Oblivious DNS [25] is a recent proposal that introduces indirection to the DNS lookup process as a way to ensure that no monitoring point can figure out both (*i*) the hostname being looked up and (*ii*) who is asking for the hostname. Another privacy issue is that queries often needlessly include an entire hostname which unnecessarily exposes potentially sensitive information to the authoritative nameservers. For instance, a query sent to a root nameserver

[6] For example, according to *root-servers.org*, UDP queries accounted for 96.2% of the queries on April 11, 2018—the day of the DITL collection we use above—and 93.4% of queries on April 11, 2019.

for "www.sensitive-domain.com" reveals a client's final target to the root nameserver even though the only part of the query the root nameserver can act upon is the final ".com" label. "QNAME minimization" (QMin) [7] has been proposed as a method to send only the germane part of a hostname to a given nameserver. QMin's use has been measured to be modest [11]—although this is expected since QMin is a relatively new approach. Rather than limiting or obscuring names as they transit the network, our approach aids privacy by eliminating the need for some transactions. As with the security benefits above, our approach does not solve the DNS privacy problem, but it does reduce the issue.

Complexity Reduction: Eliminating root nameservers reduces the complexity of some aspects of the DNS ecosystem.[7] Obviously, eliding nearly 1K servers reduces the complexity of administering critical machines and making sure they are answering DNS queries with the correct information. Another example of complexity reduction impacts recursive resolvers. When a recursive resolver needs to contact a root nameserver it must determine which of the 13 root nameservers to contact. Resolvers use a process that involves leveraging multiple roots, measuring the delay in obtaining a response and retaining a history of these measurements. This history is then used to guide future queries to the root nameserver that is likely to provide the quickest answer. Using our approach the question of which root nameserver to leverage becomes moot and the attendant complexity used to answer the question can be removed.

5 COST CONSIDERATION

We now turn to the costs of eliminating root nameservers.

5.1 Size

Eliminating root nameservers means that recursive resolvers will bootstrap with the root zone file instead of the root hints file. In some sense, this is not a fundamental change, but just a change in *which* file we use to start the lookup process. However, as we note in §2.1, the root hints file has 39 entries, while the root zone file has 22K entries—an increase of over 581x. While this increase seems stark, we believe it is manageable for several reasons.

- First, we note that 22K records is not an onerous amount of data for a modern server to manage.
- As we note in §3, a recursive resolver could place all the records from the root zone file in the cache. We took a snapshot of ICSI's[8] recursive resolver cache at 11am on June 7, 2019 and found roughly 55K RRsets being cached—including RRsets for about 20% of the TLDs. The root zone file from the same day contains just under 14K RRsets. Adding the 80% of the RRsets from the root zone file that are not already in the cache represents a roughly 20% increase in the cache size.
- Additionally, we note that previous work shows that across time and vantage point 51–86% of DNS lookups are used only once [1, 15, 28]. The resulting records do not benefit from caching and, hence, pollute a recursive resolver's cache. Therefore, even if a recursive resolver pulls all TLD records into the cache and this causes some LFU-like evictions of other names, the cache hit rate is unlikely to be impacted.
- Finally, as we discuss in §3, there is no requirement that all 22K records in the root zone file be stored in the recursive resolver's cache. Rather, the records could be read as needed from the root zone file. In this case, the caching requirement of our approach is the same as when obtaining TLD records from the root nameservers. Further, as a simple test wrote a Python script to extract all records related to a given TLD from the standard compressed root zone file. Over 1,000 trials the script takes an average of 37 msec to extract all records that pertain to a random TLD. This is similar to network round-trip times and so even a rudimentary scheme should not slow DNS lookups. Finally, there are clearly additional steps that would make the process faster—e.g., loading the root zone into a database or creating a single file for each TLD.

Given the above, while eliminating root nameservers will increase the size of a recursive resolver's bootstrapping information, we believe the increase is manageable.

5.2 Distribution Load

By eliminating root nameservers we add a requirement for recursive resolvers to download the root zone file periodically. On June 7, 2019 the compressed root zone file was approximately 1.1 MB. Each recursive resolver will need to download the file approximately every two days given the TTLs within the zone file (see §2.1). This is not a large distribution requirement for modern networks. As we note in §3, we can leverage a variety of distribution models—from mirrors to peer-to-peer systems—to distribute the file.

As a point of comparison, ICSI uses a series of SpamHaus blacklists [29] as part of our email processing. We run an rsync to one of SpamHaus' servers every minute to retrieve the changes in the blacklists. On May 20, 2019 these downloads totaled 3.1 GB. Our operators consider this a not-too-high cost of running the network. The email blacklist is just an example. Modern systems and networks download much configuration information that is at least on the order of the

[7] See §5.4 for a sketch of the complexity our proposal introduces.

[8] ICSI is modest in size—with on the order of 100 active local and remote users at any given time during the business day.

size of the root zone file—from routing tables to anti-virus databases to IDS signatures.

One way to further mitigate the distribution requirement is to increase the TTLs in the root zone file. Of course, increasing TTLs comes with a cost in terms of flexibility. As an initial analysis we aim to understand whether increasing the TTLs would have a practical impact. We analyzed a snapshot of the root zone file from each day in April, 2019. On the first of the month the root zone included 1,532 TLDs and one was deleted during the month. Of the TLDs, all but five have at least one nameserver (by IP address) that is constant for the entire month. That is, if a recursive resolver used a root zone file that was out of date by one month, 99.6% of the TLDs would remain accessible. The five TLDs that do not have a constant nameserver for the entire month are run by NeuStar and use a slowly rotating set of IP addresses for the TLD nameservers. The overlap ensures that a root zone file that is no more than 14 days out of date will ensure constant TLD reachability. Further, comparing the root zone files on April 1, 2018 and April 1, 2019 we find that all but 50 TLDs (3.3%) would still retain reachability with a root zone file that is a year out of date. We do not advocate using out of date zone files. However, this analysis shows that the contents of the zone files are highly stable and the TTLs could be increased (e.g., to 1 week). In turn, this would reduce the root zone file distribution overhead.

Given the above analyses, we believe the load imposed by distributing root zone files is not an impediment to eliminating root nameservers and in fact could be reduced further with a modest increase in the TTLs.

5.3 TLD Additions

A downside of downloading a copy of the root zone file periodically is that the contents remain static between fetches. Hence, if a TLD is added to the root zone there will be a lag before a recursive resolver's clients can access that TLD. Further, the longer we extend the TTL—per §5.2—the longer the average lag before new TLDs are available. To put this lag in context we analyze lookups to *j*-root for the ".llc" TLD, which was the last TLD added before the DITL 2018 data collection. The ".llc" TLD was added on February 23, 2018 and our dataset is from 47 days later on April 11, 2018. We find that of the nearly 5.7B requests to *j*-root only 6.5K (< 0.0002%) are for the ".llc" TLD. Further, of the 4.1M recursive resolvers sending requests to *j*-root, only 1,817 (< 0.1%) request the ".llc" TLD. This shows that over six weeks after ".llc" was added the TLD remained unpopular. While only an anecdote, this suggests that even if TTLs are expanded beyond two days it seems unlikely that new TLDs present a large issue. Further, if this is deemed a large issue, we could augment the root zone file with a small "recent additions" or "diffs" file to allow resolvers to cheaply and fairly constantly obtain information about newly added TLDs.

5.4 Recursive Resolver Complexity

In §4 we sketch how some aspects of the DNS ecosystem get less complex by eliminating the root nameservers. However, the reverse is also true. We note two forms of additional complexity here:

- By requiring recursive resolvers to download a copy of the root zone file and incorporate the resulting information in the lookup process the resolvers get more complicated. We believe this complexity can be built into the resolver software and while operators will need to be aware of it, the task can be automated and abstracted away from any direct intervention. Further, this general approach of retrieving and loading new configurations into an operational system is already widely used and accepted within the operations community. Examples of this include downloading new anti-virus databases, blacklists and IDS signatures.
- A distribution mechanism for root zone files must also be setup. As a community we have much relevant experience in distributing files—including software, apps, songs, movies, etc.—in a robust fashion. Therefore, we expect the distribution of the root zone file to be straightforward.

While our approach adds complexity to some parts of the DNS ecosystem, the nature of the complexity is not deep or novel, but rather requires a set of straightforward changes.

6 RELATED WORK

The closest related work to our position is RFC 7706 [22]. The RFC does not suggest eliminating root nameservers, but rather sketches why and how to augment the traditional root nameservers with a local instance as an optimization. That is, the local instance of the root nameserver is for getting answers more quickly and reliably than using one of the standard root nameservers. Our position, on the other hand, is that root nameservers should be eliminated as their cost far outweighs their benefits at this point. RFC 7706 nicely illustrates the practical ways organizations can operate without community-provided root nameservers.

Our position is somewhat similar to the "DNS push" proposal [14]. DNS push augments the current DNS by making recursive DNS resolvers part of a peer-to-peer system that is used to replicate the nameserver records found in the root and TLD zone files. The aim of broadly replicating nameserver information is to reduce the reliance on a small number of authoritative servers that can be subject to attack. Subsequent to the DNS push proposal, the root nameservers have been broadly replicated via anycast instead of using the

proposed peer-to-peer system. However, the general strategy of broad replication has proven to provide for robust service. Our proposal is simultaneously more and less ambitious than DNS push. While not based on new peer-to-peer system, our position calls for even broader replication of the root zone file than DNS push. However, in trade for replication breadth we suggest replicating a more narrow set of records—i.e., only the root zone file.

Our position can be seen as a subset of sorts of the concepts from the Grapevine system [6]. Grapevine groups individual names into "registries" which are akin to DNS zones. These registries are replicated in the system, but only as whole units and not as individual names. We are not advocating replicating all zones, but only the root zone.

There are many DNS measurement studies in the literature. As we refer to above, the most related to our work are those that consider the use of the root nameservers [8, 9, 23, 31]. In this paper we provide simple measurements that suggests these previous—and more detailed—studies still hold.

Finally, we note that we are not the first to suggest removing a piece of the DNS ecosystem. Schomp, et.al. [26] observes that recursive resolvers and DNS forwarders are often at fault for privacy and security issues and therefore suggest removing these components from the lookup process and instead charging client devices with directly interacting with the authoritative infrastructure to conduct DNS lookups.

7 CONCLUSION

The general trend in networks is to make them more complicated. We believe that we have demonstrated that the time is nigh to begin a thinking about the efficacy of continuing to operate a traditional set of root nameservers. The cost of this infrastructure is high and increasing. Meanwhile, the benefits compared to an alternate arrangement where recursive resolvers use a local copy of the information are low. Our hope is that our analysis of the question will spur a conversation within the community.

ACKNOWLEDGMENTS

Albert Park helped us understand ICSI's DNS cache usage, as well as our operational SpamHaus setup. Craig Partridge, Michael Rabinovich and the anonymous reviewers provided valuable comments on a draft of this paper. We thank *j*-root for the data used in this paper, as well as DNS-OARC for facilitating the DITL data collection and providing an analysis platform. The work in this paper was funded in part by NSF grants CNS-1237265, CNS-1647126, CNS-1518918 and CNS-1815876. Thanks to Sally Floyd for influencing this work—and the author—in countless ways.

REFERENCES

[1] Mario Almeida, Alessandro Finamore, Diego Perino, Narseo Vallina-Rodriguez, and Matteo Varvello. 2017. Dissecting DNS Stakeholders in Mobile Networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*.

[2] Andrew Tridgell and Paul Mackerras. [n. d.]. rsync. ([n. d.]). https://rsync.samba.org/.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. DNS Security Introduction And Requirements. RFC 4033. (March 2005).

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. Protocol Modifications For the DNS Security Extensions. RFC 4035. (March 2005).

[5] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. Resource Records For the DNS Security Extensions. RFC 4034. (March 2005).

[6] Andrew Birrell, Roy Levin, Roger Needham, and Michael schroeder. 1982. Grapevine: An Exercise in Distributed Computing. *Communications of the ACM* 25, 4 (April 1982).

[7] S. Bortzmeyer. 2016. DNS Query Name Minimisation To Improve Privacy. RFC 7816. (March 2016).

[8] N. Brownlee, k. claffy, and E. Nemeth. 2001. DNS Measurements at a Root Server. In *IEEE Global Telecommunications Conference (GLOBECOM)*.

[9] S. Castro, D. Wessels, M. Fomenkov, and k. claffy. 2008. A Day at the Root of the Internet. *ACM SIGCOMM Computer Communication Review (CCR)* 38, 5 (Oct 2008).

[10] D. Dagon, N. Provos, C.P. Lee, and W. Lee. 2008. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In *NDSS*.

[11] W.B. de Vries, Q. Scheitle, M. Müller, W. Toorop, R. Dolmans, and R. van Rijswijk-Deij. 2019. A First Look at QNAME Minimization in the Domain Name System. In *Passive and Active Measurement Conference*.

[12] S. Dickinson, D. Gillmor, and T. Reddy. 2018. Usage Profiles For DNS Over TLS And DNS Over DTLS. RFC 8310. (March 2018).

[13] Domain Name System Operations Analysis and Research Center. [n. d.]. ([n. d.]). http://www.dns-oarc.net/.

[14] Mark Handley and Adam Greenhalgh. 2005. The Case for Pushing DNS. In *ACM SIGCOMM HotNets*.

[15] Shuai Hao and Haining Wang. 2017. Exploring Domain Name Based Features on the Effectiveness of DNS Caching. *ACM SIGCOMM Computer Communication Review* 47, 1 (2017).

[16] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In *IEEE Communications and Network Security*.

[17] P. Hoffman and P. McManus. 2018. DNS Queries Over HTTPS (DoH). RFC 8484. (Oct. 2018).

[18] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. Specification For DNS Over Transport Layer Security (TLS). RFC 7858. (May 2016).

[19] ICANN DNS Zone Transfer. [n. d.]. ([n. d.]). https://www.dns.icann.org/services/axfr/.

[20] Ben Jones, Nick Feamster, Vern Paxson, Nicholas Weaver, and Mark Allman. 2016. Detecting DNS Root Manipulation. In *Passive and Active Measurement Conference*.

[21] D. Kaminsky. 2008. Black Ops 2008: It's the End of the Cache As We Know It. *Black Hat USA* (2008).

[22] W. Kumari and P. Hoffman. 2015. Decreasing Access Time To Root Servers By Running One On Loopback. RFC 7706. (Nov. 2015).

[23] Matthew Lentz, Dave Levin, Jason Castonguay, Neil Spring, and Bobby Bhattacharjee. 2013. D-mystifying the d-root Address Change. In *ACM Internet Measurement Conference*.

[24] P.V. Mockapetris. 1987. Domain Names - Concepts And Facilities. RFC 1034. (Nov. 1987).

[25] P. Schmitt, A. Edmundson, and N. Feamster. 2019. Oblivious DNS: Practical Privacy for DNS Queries. In *Privacy Enhancing Technologies Symposium*.

[26] Kyle Schomp, Mark Allman, and Michael Rabinovich. 2014. DNS Resolvers Considered Harmful. In *ACM SIGCOMM HotNets*.

[27] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2014. Assessing DNS Vulnerability to Record Injection. In *Passive and Active Measurement Conference*.

[28] Kyle Schomp, Michael Rabinovich, and Mark Allman. 2016. Towards a Model of DNS Client Behavior. In *Passive and Active Measurement Conference*.

[29] Spamhaus DNS Blacklists. [n. d.]. ([n. d.]). http://www.spamhaus.org/.

[30] Andrew Tridgell. 1999. *Efficient Algorithms for Sorting and Synchronization*. Ph.D. Dissertation. The Austrailian National University.

[31] D. Wessels and M. Fomenkov. 2003. Wow, That's A Lot of Packets. In *Passive and Active Network Measurement Workshop (PAM)*.

[32] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-Oriented DNS to Improve Privacy and Security. In *IEEE Symposium on Security and Privacy*.