**The ISP Column**
*A monthly column on things Internet*

September 2020
Geoff Huston

# Scaling the Root of the DNS

The DNS is a remarkably simple system. You send it queries and you get back answers. Within the system you see exactly the same simplicity: The DNS resolver that receives your query may not know the answer, so it, in turn, will send queries deeper into the system and collects the answers. The query and response process is the same, applied recursively. Simple.

However, the DNS is simple in the same way that Chess or Go are simple. They are all constrained environments governed by a small set of rigid rules, but they all possess astonishing complexity.

Simple systems can have very deep complexity. This is a major theme in the study of Formal Systems.

The study of mathematics in the 19th century moved into to dizzying heights of self-reflection. The question they were trying to answer was: What are the formal assumptions upon which the entire superstructure of mathematics was constructed? The Peano axioms for natural numbers are a good example here. If you took these axioms and only applied operations from a constrained and well-defined set, then was it possible to derive every known truth (provably correct) statement in maths? This question motivated Whitehead and Russell to labour on the landmark three volume work Principia Mathematica in the early 20th century, that was intended to build the entire edifice of mathematics using only first principles and symbolic logic. Their work was in part brought about by an interest in logicism, the view on which all mathematical truths are logical truths. Mathematics was seen as a "pure" form of philosophical study, whose truths were independent of any observer or any natural system. This study led to work by Kurt Gödel that probed the limits of this approach. His *Incompleteness Theorems* are two theorems of mathematical logic that demonstrate the inherent limitations of every formal axiomatic system capable of modelling basic arithmetic. These results are important both in mathematical logic and in the philosophy of mathematics. The first incompleteness theorem states that no consistent system of axioms whose theorems can be listed by an effective procedure is capable of proving all truths about the arithmetic of natural numbers. For any such consistent formal system, there will always be statements about natural numbers that are true, but that are unprovable within the system. The second incompleteness theorem, an extension of the first, shows that the system cannot demonstrate its own consistency.

With all our faith in rule-based automated systems that largely operate today's digital world its sobering to realize that the limitations of such a world view were clearly stated by Kurt Gödel 1931.

Why am I talking about this? Well, the informal expression of Gödel's work is that any formal system that is powerful enough to be useful is also powerful enough to express paradoxes. Or more informally, any constrained simple system that is sufficiently useful to express compound concepts is also capable of impenetrable complexity. And this is where the DNS comes in!

## The Root Zone

The DNS is not a dictionary of any natural language, although these days when we use terms like "facebook.com" as proper nouns we may be excused from getting the two concepts confused! The DNS is a hierarchical name space. Domain Names are constructed using an ordered sequence of labels. This ordered sequence of labels serves a number of functions, but perhaps most usefully it can be used as an implicit procedure to translate a Domain Name into an associated attribute through the DNS name resolution protocol.

For example, I operate a web server that is accessed using the DNS name *www.potaroo.net*. If you direct your browser to this DNS name then your browser firstly needs to translate this DNS name to an IP address, so that your browser knows where to send the IP packets to perform a transaction with my server. This is where the structure of the name is used. In this case the DNS system will query a root server to translate this name to a corresponding IP address and the response will be the set of resolvers that are authoritative for the *.net* zone. Ask any of these *.net* servers for this same name and the response will be the servers that are authoritative for the *potaroo.net* zone. Ask any of these *potaroo.net* servers for the same name and you will get back the IP address you are looking for. Every DNS name can be decomposed in the same way. The name itself defines the order of name resolution processing.

There is one starting point for every DNS resolution operation: the root zone.

There is a school of thought that decries any exceptional treatment given to the root zone of the DNS. It's just another zone, like any other. Its set of authoritative servers receive queries and answer them, like any other zone. There's no magic in the root zone and all this attention is entirely unwarranted.

However, I think this understates the importance of the root zone in the DNS. The DNS can be seen as a massive distributed database. Indeed, it's so massive that there is no single static map that identifies every authoritative source of information and the collection of data points about which it is authoritative. Instead we use a process of *dynamic discovery* where the resolution of a DNS name firstly is directed to locating the authoritative server that has the data relating to the name we want resolved, and then querying this server for the data. The beauty of this system is that these discovery queries and the ultimate query are precisely the same query in every case.

But everyone has to start somewhere. A DNS recursive resolver does not know all the DNS' authoritative servers in advance and never will. But it does know one thing. It knows the IP address of at least one of the root servers. From this starting point everything can be constructed on the fly. The resolver can ask a root server for the names and IP addresses of all other root servers (the so-called *priming query*), and it can store that answer in a local cache. When the resolver is given a name to resolve it can then start with a query to a root server to find the next point in the name delegation hierarchy and go on from there.

If this was how the DNS actually worked then it's pretty clear that the DNS system would've melted down in a few seconds. What makes this approach viable is *local caching*. A DNS resolver will store the answers in a local cache and use this locally held information to answer subsequent queries for the life of the cached entry. So perhaps a more refined statement of the role of the root servers is that every DNS resolution operation starts with a query to the cached state of the root zone. If the query cannot be answered from the local cache, then a root server is queried.

However, behind this statement lurks an uncomfortable observation. If the root servers are inaccessible, then the entire DNS ceases to function. This is perhaps a dramatic overstatement in some respects, as there would be no sudden collapse of the DNS and the Internet along with it. In the hypothetical situation where all the instances of the root servers were inaccessible then DNS resolvers would continue to work using locally cached information. However, as these cache entries timed out, they would be discarded from these local resolvers as they could not be refreshed by re-querying the root servers. The lights in the DNS would fade to black bit by bit as cached entries timed out. For that reason, the DNS root zone is different from every other zone. It's the zone that is the master lookup for every other zone. That's why it deserves particular attention.

Due to local caching, root zone servers are not used for every DNS lookup. The theory is that the root servers will only see queries as a result of cache misses. With a relatively small root zone and a relatively small set of DNS resolvers then the root zone query load should be small. Even as the Internet expands its user base the query load does not necessarily rise. It's the number of DNS resolvers that determines root server query load if we believe in this theory of the root's operation in the DNS.

However, the theory does not hold up under operational experience. Why do we see a continuing pattern of growth of queries seen by the collection of root servers? The total volume of queries per day recorded by the Root servers is shown in Figure 1.

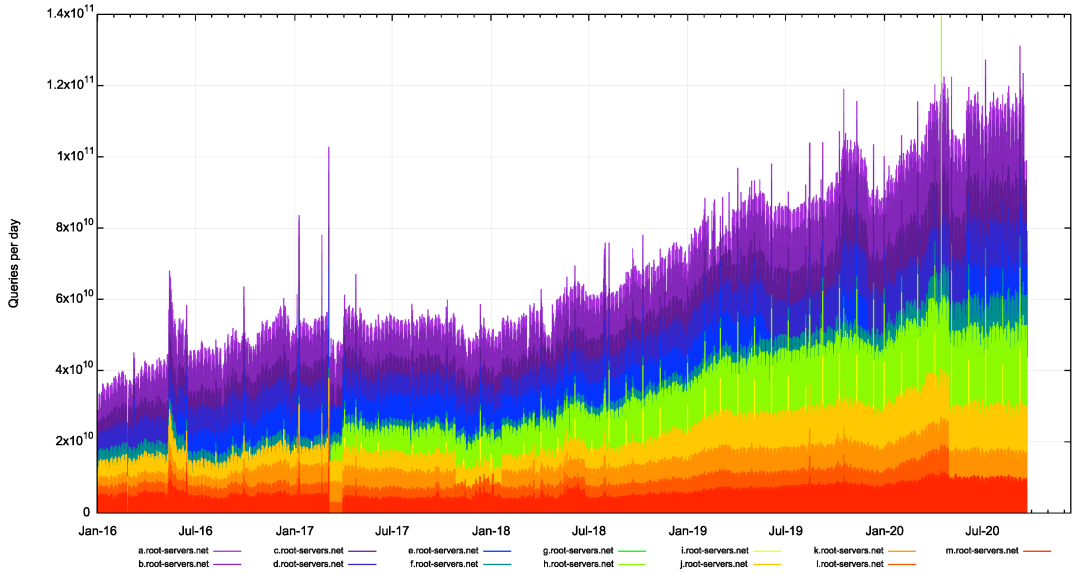


*Figure 1 – Total Root Servers Queries per Day (RSSAC002 data)*

Over the past 4 years the volume of queries seen by the collection of root servers appears to have tripled.

What are we doing in response?

> The data published by the root servers uses the framework published in RSSAC002, generated independently by most of the root service operators. (The term "most" means that I can't locate data from B, E or G roots for total query counts and A, B, E, F, G or J roots, for daily response code counts. The publication rate is not exactly daily either).

What can we say about the root service as a whole? Nothing that is complete is the frustrating answer. We get a piecemeal glimpse into the root system and it's not clear to what extent the data that is published is consistent over time and not clear to what extent the data that is published is related to the whole. Does the published data present 70% of the whole? Or 95%? Or something in between? Nobody knows. In making the assertion that the query volume over the past 4 years "appears to have tripled" I'm taking a big liberty with this rather incomplete data set.

This is a pity, because without a solid foundation of data its challenging to make some important judgements about the characteristics of the root system. For example, how has NSEC caching affected root query volumes? Or the use of Local Root? How quickly is the query volume growing? Why is it growing? How can we respond?

Like many things in our world we get what we pay for. The root service is a free service provided on the basis of altruism rather than as a contracted funded service. So perhaps with this rather noisy and incomplete data set we're already getting more than what we are paying for!
https://www.icann.org/en/system/files/files/rssac-002-measurements-root-06jun16-en.pdf
https://root-servers.org

## Root Zone Scaling

The work to increase the capacity of the root zone is a never-ending task. Figure 1 shows that the 60 billion queries per day seen by most of the root servers in mid-2018 has grown to 120 billion queries per day in mid-2020. How are the root server operators responding?

The first set of responses to these scaling issues was in building root servers that had greater network capacity and greater processing throughput. But with just 13 servers to work with this was never going to scale at the pace of the Internet and we needed something more. The next scaling step has been in the conversion from unicast to anycast services. There may be 26 unique IP addresses for root servers (13 in IPv4 and 13 in IPv6) but each of these service operators now use *anycast* to replicate the root service in different locations. The current number of root server sites is described at root-servers.org (Table 1)

| Root | Anycast Sites |
|---|---|
| A | 16 |
| B | 6 |
| C | 10 |
| D | 149 |
| E | 254 |
| F | 242 |
| G | 6 |
| H | 8 |
| I | 64 |
| J | 118 |
| K | 73 |
| L | 147 |
| M | 5 |

*Table 1 – Anycast Sites for Root Servers*

That's a total of 1,098 sites where there are instances of root servers.

The number of server engines is larger than that count of the number of sites, as its common these days to use multiple server engines within a site and use some form of query distribution front-end to distribute the incoming query load across multiple back-end engines

However even this form of distributed service engineering may not be enough. In two years from now we may need double the root service capacity from the current levels, and in a further two years we'll need to double it again. And again and again and again. Exponential growth is a very harsh master.

However, the task is perhaps even more challenging that simple scaling. We also need to think about money.

## A Vestige of Altruism in a Venal Internet

The model used by the Root Service is undoubtedly an anachronism in today's Internet.

The 12 organizations who operate an instance of the root servers do so without any form of direct payment or compensation. When the IANA first enrolled organizations to undertake this role, it was in an environment where the Internet was largely a research project making its first steps into a global service. The selection of root server operators was based on a desire to locate servers in a pattern that matched the user population of the time, and in a largely autonomous mode where no central funding obligations were offered. The undertaking to operate a root service was based at the time on the understanding that there was no funding to undertake the role.

That was more than 30 years ago, and the Internet has changed dramatically in so many ways since then. However, one thing has not changed. There is still no organized funding model for the root service. The administrators of the top-level domains that are listed in the root zone don't directly or indirectly pay for the infrastructure to support root zone queries. The various resolvers that pass queries to the root zone servers, don't directly or indirectly pay either. Nobody pays. It's still a service operated on an altruistic basis.

The capacity provided by the aggregate of the root service operators needs to double every two years or so, but it has to do so without any common or structured funding arrangements. And unless something else changes in the way we use the DNS this doubling looks like it will continue, so the task of fielding capacity to meet this query load just gets more expensive over time.

However, let me add that this is certainly not an insurmountable problem and there is no immediate crisis at hand. The root service is still amply capable of both meeting current query loads and absorbing most forms of DDOS attacks that get directed against it. However, such an approach of continued over-engineering this service seems to be more along the lines of a brute force saturation response that demands ever-increasing capacity. Perhaps there are other approaches that might mitigate the amount of traffic passed to the root servers? What do we know about these queries that are being passed to the root? Could different approaches mitigate some of this traffic?

To see how such a question could be answered it may be useful to look at the query traffic that is passed to the root servers.

## Root Queries

There have been many studies of the root service and the behavior of the DNS over the past few decades. If the root servers were simply meant to collect the cache misses of DNS resolvers, then whatever is happening at the root is not entirely consistent with a model of resolver cache misses. Indeed, it's not clear what going on at the root!

It has been reported that the majority of queries to the root servers result in NXDOMAIN responses. In looking at the published response code data, it appears that some 75% of root zone queries result in NXDOMAIN responses (Figure 2), and this relative proportion has been growing in the past couple of years.

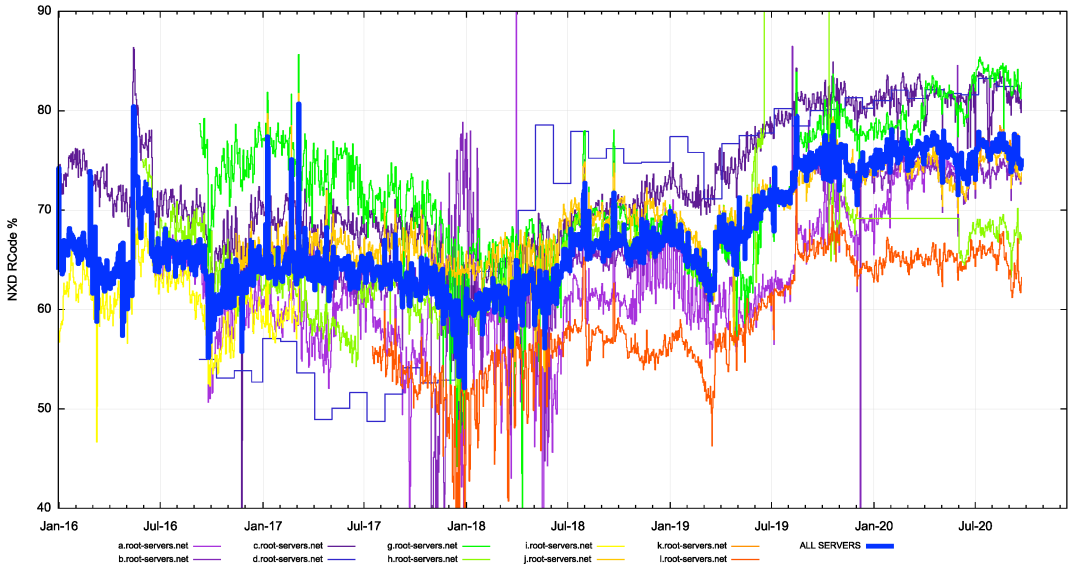


*Figure 2 – Proportion of Root Zone NXDOMAIN responses per Day (RSSAC002 data)*

There is a very curious aspect of this behaviour, in that the percentage of query traffic seen by each of the root service letters appears to vary significantly (by up to 20%). In many respects the root servers are intentionally identical, and it is unusual to see such variation in the root zone query profiles from each root service.

As Versign's Duane Wessels reported to the DNSOARC 32 meeting in June 2020, the Chrome browser generates three single label DNS queries, where the label is between 7 to 15 characters in length and composed of alpha characters. Prior to February 2015 the code used by this browser generated only 10-character labels, and the switch to randomized lengths as well as randomized labels was made in code releases as of February 2015. The browser engine makes thise queries at startup, and also when the local IP address changes and if the local DNS server changes.

The motivation for the Google's Chrome browser to do this is pretty obvious. Many ISPs perform NXDOMAIN substitution in their DNS resolvers and replace the "no such domain" response with a synthetic pointer to their own search page, which allows them to monetize all those No such domain" user typos. From Google's perspective this NXDOMAIN substitution is not well regarded. Search is a major path to advertisement placement and advertising is Google's core revenue. So how can Google's Chrome browser platform detect network environments where NXDOMAIN substitution is happening? Simple. Test the DNS with its own nonce random string queries to see if NXDOMAIN substitution is taking place.

This supposedly innocuous probe of the DNS should've been pretty harmless. But there is a lot of Chrome out there these days. Some two thirds of all browser use in today's Internet is reported to use the Chrome browser, and the number rises when you include products such as Edge that are based on the Chrome engine. Consequently, it is perhaps unsurprising to learn that according to this report these probes are now taking up some 50% of the total root server traffic (Figure 3)

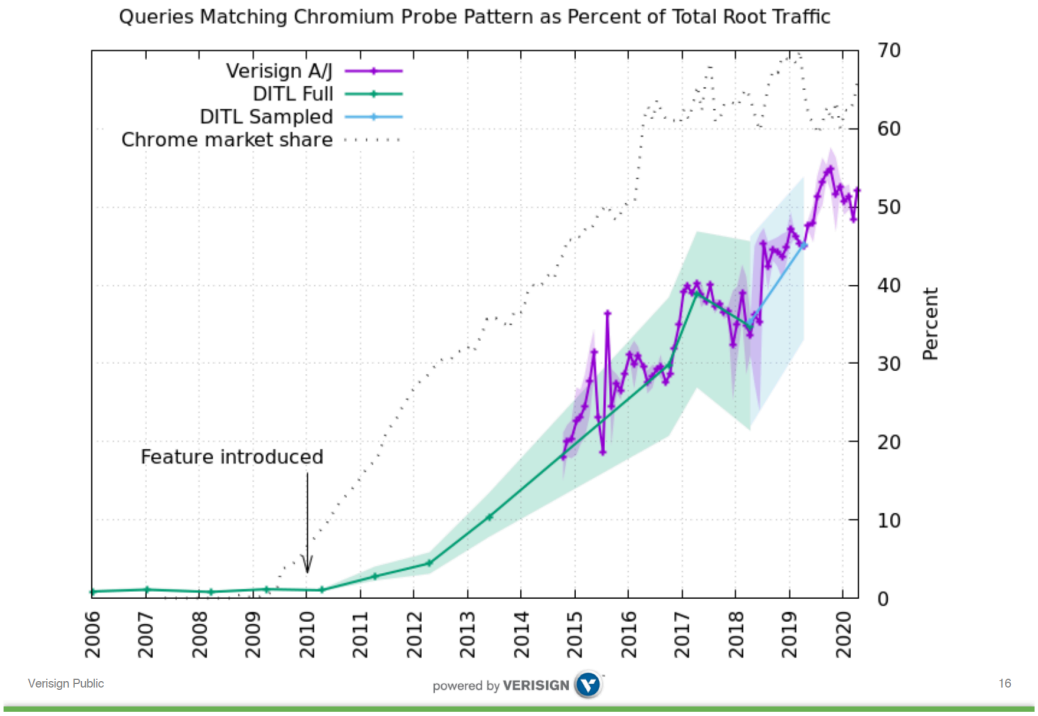


*Figure 3 - Chrome queries see at the Root. From "Intranet Redirect Detector or Pseudo Random Subdomain Attack?", Duane Wessels, Verisign, June 2020*
https://indico.dns-oarc.net/event/35/contributions/767/attachments/743/1260/OARC32a-chromium_queries_root.pdf

Part of the strength of the Internet lies in the decoupled nature of the network's infrastructure, where many component service providers operate within their chosen niche of activity, and the overall orchestration of the collective efforts is left to market forces. No one is in charge. But while this is a strength it can also be a weakness, particularly in cases of cost displacement. The design decision by Chrome to probe for NXDOMAIN substitution through one-off labels queries is a decision that imposes negligible marginal cost to Chrome or Chrome users. However, it does impose significant costs to root service operators given that one half of their overall query load come from this behaviour.

But in the same way cost and benefit are displaced, the tools to remedy this situation lie in the hands of a third class of actors. If all recursive resolvers, and their front-end load balancers, performed effective NSEC caching (and presumably DNSSEC validation as well) then these random non-existent top-level label Chrome queries would be absorbed by the NSEC cache in the recursive resolver.

In a centrally orchestrated environment, the costs and benefits could be directly compared, and such solutions could be deployed where it was cost-beneficial to do so. Without such orchestration there is little in the way of incentive for either the Chrome group within Google, or the recursive resolver operators to spend their time and effort to address how to mitigate this class of queries, so the root servers are left with the problem without the means of providing incentives for any other party to provide a remedy.

However, it may be unfair to attribute the entirety of this query traffic directly to Chrome. When we looked into the way NXDOMAIN responses are handled in the DNS, we found that the DNS itself was part of the problem. We looked at the behaviour of the DNS when a browser passed a query to the DNS where the response was NXDOMAIN, and performed this test across some 7 million users across the Internet (https://www.potaroo.net/ispcol/2019-02/nxd.html). At the authoritative servers for this domain name we saw an average of 2.2 queries per original browser query, more than double what we might have

naively expected. So perhaps Chrome only contributes 25% of the load at the root servers, and the DNS resolver infrastructure is responsible for doubling the query volume presented to the root servers.

There is an additional amplification factor here. The DNS also has a sizeable proportion of 'zombie' traffic. A study by APNIC labs using a dynamic DNS label that included the label creation time found that around 25% of all DNS queries seen at our authoritative server were queries that were unrelated to the original use of this DNS label (https://www.potaroo.net/ispcol/2016-03/zombies.pdf). There are a number of reasons why "old" queries are replayed in the DNS. There are many points where queries are captured and logged, and analysis of these logs appear to involve re-query. In other cases the resolver appears to get stuck in a tight code loop and makes a huge number (billions of queries over weeks) of repeat queries for the same DNS name.

While it's not all directly due to Chrome browser engines, if Chrome were to perform this query using a zone deeper in the DNS hierarchy than the root zone, the original query load and the amplified load would drop from the root zone. The root servers would shed more than half of their current query load.

As well as this significant traffic component, there are also factors of name leakage. Many environments use locally defined DNS zones to label services, and when devices are moved out of these local domains they may still query for these locally defined names. Queries at the root for undelegated zones include the top level labels .home, .corp, .local, and .mail, as well as a number of common CPE vendor names (https://www.potaroo.net/presentations/2014-06-24-namecollide.pdf). Delegation of these labels would push such queries away from the root zone, but at the same time might open up a set of security issues where names intended to be resolved in one network context were then resolved in a different context, with surprising and potentially compromising outcomes. While there are individual cases where such leaks of queries to the root can be mitigated by altering the behaviour of an application or device through field updates, in other cases the behaviour is more deeply entrenched and more difficult to stop.

## Query Deflection

Altering the behaviour of devices and applications to use delegated domains even in private contexts to push queries away from the root is one possible approach to mitigate the exponential growth in query volumes at the root. I'm not overly optimistic that this would be very effective, as the current cost allocations in the DNS work against this. Using the root is the fast and free option, and the root servers are very attentive to data privacy. Using non-delegated private use domains or random names in the case of Chrome is an option that is without cost and the consequent query data is largely private. Any alternative approach is going to push the load to other servers, and that might expose a cost to the application or device vendor. Using the root zone is free, fast and it just works! What's the problem?

So perhaps we need to look at other approaches. How else can we deflect these queries away from the root server system?

There are two approaches that may help.

### NSEC Caching

The first is described in RFC8198, or NSEC caching. When a top level label does not exist in a DNSSEC-signed zone, and the query has the DNSSEC EDNS(0) flag enabled, the NXDOMAIN response from a root server includes a signed NSEC record that gives the two labels that do exist in the root zone and "surround" the non-existent label. NSEC records say more than "this label is not in this zone". It says that every label that is lexicographically between these two labels does not exist. If the recursive resolver caches this NSEC record it can use this same cached record to respond to all subsequent queries for names in this label range, in the same way that it conventionally uses "positive" cache records.

If all recursive resolvers performed NSEC caching then the query volumes seen at the root from recursive resolvers, including those associated with Chrome queries, would vanish.

So NSEC caching is important for recursive resolvers is important. Bind supports this as of release 9.12. Unbound supports this as of release 1.7.0. Knot resolver supports this as of 2.0.0.. But the queries at the root zone keep growing.

At APNIC Labs we set up a measurement of NSEC caching, and reported the results of this effort at a DNS OARC meeting in October 2019 (https://www.potaroo.net/presentations/2019-10-31-oarc-nsec-caching.pdf). The exercise was not exactly heartening. We used a methodology that used two queries, where the first query generated a NXDOMAIN response and a NSEC record if the resolver had the DNSSEC flag set in the query, then waited for 2 seconds and performed a second query into the NSEC range. In theory we should see the first and not see the second query. Some 30% of users sit behind DNS resolvers that set the DNSSEC flag in queries and are observed to perform DNSSEC validation, so we might expect that such an experiment would reveal an uptake level of NSEC caching for 30% of users. What we observed was a far lower measurement of 7% of users (Figure 4)

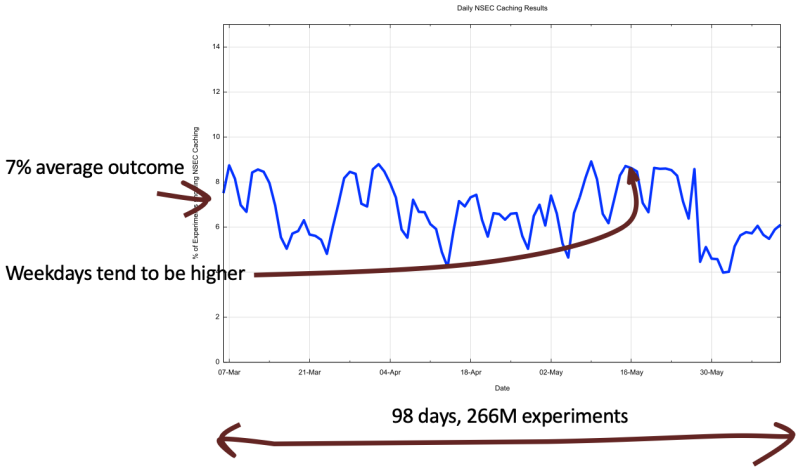


*Figure 4 – Measurement of NSEC caching, April 2019*

It might have just been too early. When we measured Query Name minimization in August 2019, we saw a 3% deployment and a year later in August 2020 the same measurement showed a 18% deployment. So perhaps we were impatient and measures too early, and if we repeated the measurement today the number might be higher. However, it's also the case that the measurement technique was not well attuned to the DNS infrastructure model used today. These days much of the DNS resolver infrastructure sits in "farms" where a front-end query distributor passes each incoming query to one of a collection of DNS resolver engines within the back-end farm. While the front end might try to optimize cache performance for queries for the same domain name and direct all such queries to the same engine, the same approach would not necessarily be effective for name ranges and queried for different labels might be directed to different resolver engines. And lastly there is the possibility that NSEC caching might well be working already! The relative proportion of NXDOMAIN responses has remained steady at 75% of all responses for the past 12 months, and the total root query volume has remained relatively steady for the past 4 months. Maybe NSEC caching is already working (Unfortunately the caveats about the quality and consistency of root server reports apply here and we can do little more than speculate without much in the way of solid data to justify the speculation.) We just don't know from the available data. NSEC caching could be working already in the DNS, and continued uptake of this behaviour in DNS resolvers might generate further results in coming months in both a decreasing proportion of NXDOMAIN responses and decrease of overall query volumes.

However, NSEC caching is a tactical response to root zone scaling concerns, as distinct from a strategic response. It's still dependent on the root server infrastructure and uses a query-based method of promulgating the contents of the root zone. Nothing really changes in the root service model. What NSEC caching does is allow the resolver to make full use of the information in the NSEC response. Nothing else changes.

### Local Root

Another option is to jump out of the query/response model of learning the contents of the root zone and simply load the entire root zone into recursive resolvers. The idea is that if a recursive resolver is loaded with a copy of the root zone then it can operate autonomously with respect to the root servers for the period of validity of the local copy of the root zone contents. It will send no further queries to the root servers. The procedures to follow to load a local root zone are well documented in RFC8806, and I should also note the LocalRoot service (https://localroot.isi.edu/) that offers DNS NOTIFY messages when the root zone changes.

This approach has its drawbacks at the moment. It's clumsy to use. How do you know that the zone you are serving is the current genuine root zone? Yes, the zone is signed, but not every element in the zone is signed (NS records, for example) and the client is left with the task of performing a validation of every digital signature in the zone, and at present there are some 1,376 of them. Until the root zone is signed in its entirety (a proposal that is currently still a draft in the IETF process: https://tools.ietf.org/html/draft-ietf-dnsop-dns-zone-digest-09), then you can't be sure that what you get as the root zone is the root zone.

But this model can change the nature of the root service, unlike NSEC caching. If there is one thing we've learned to do astonishing well in recent times its distribution of content. Indeed, we've conc entreated on this activity to such an extent that it appears that the entire Internet is nothing more than a small set of Content Distribution Networks. If the root zone is signed in its entirety with zone signatures that allow a recursive resolver to confirm its validity and currency and submitted into these distribution systems as just another object then the CDN infrastructure is perfectly capable of feeding this zone to the entire collection of recursive resolvers. Perhaps if we changed the management regime of the root zone to generate a new zone file every 24 hours according to a strict schedule we can eliminate the entire notification superstructure. Each iteration of the root zone contents is published 2 hours in adfvance and is valid for precisely 24 hours, for example.

## Options? Pick them all!

We operate the root service in its current guise because so far its worked adequately well. But we don't have to continue that way. At the moment we have options as to how the service can evolve.

By deflecting some of the current load to delegation points lower in the domain hierarchy we can make a massive change in the current root query load.

By having resolvers make better use of signed NSEC records we can stave off some of the more pressing immediate issues about further scaling of the root system.

But that's probably not enough. We can either wait for the system to collapse and then try and salvage the DNS from the broken mess, or maybe we could explore some alternatives now, and look at how we can break out of a query based root content promulgation model and view the root zone as just another content in the larger ecosystem of content distribution. If we can cost efficiently load every recursive resolver with a current copy of the root zone, and these days that's not even a remotely challenging target, then perhaps we can put aside the issues of how to scale the root server system to serve ever greater quantities of "NO!" to ever more demanding clients!

## Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

## Author

*Geoff Huston* B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

*www.potaroo.net*