February 2025
Geoff Huston

The Root of the DNS

The Internet's Domain Name System (DNS) is a remarkably simple system. You send queries into this system via a call to your local host's name resolution library, and you get back answers. If you peek into the DNS system you'll 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. This query/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 generate surprising complexity in their operation.

The Root Zone

The DNS is not a dictionary of any natural language, although these days when we use DNS names in our written and spoken communications, we might be excused from getting the two concepts confused! The DNS is a hierarchical name space. Individual 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 value 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 load the contents of this DNS name then your system firstly needs to resolve 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. But how does the system know which nameserver is authoritative for the zone that includes the name `www.potaroo.net`?

This is where the structure of the name space is used to discover this nameserver. In this case, the DNS resolver will query a root server to resolve this name. As this is not a name that is defined within the root zone (the zone that is server by the root servers), the response from any root server to such a query will be a referral response. In this example case, this is a redirection response, that lists the set of nameservers that are authoritative for the `.net` zone. Ask any of these `.net` nameservers for this same DNS name and again you will get back a redirection response, consisting of the list of nameservers that are authoritative for the `potaroo.net` zone. Ask any of these `potaroo.net` nameservers for the same name, `www.potaroo.net`, and you will get back the IP address you are looking for (Figure 1).

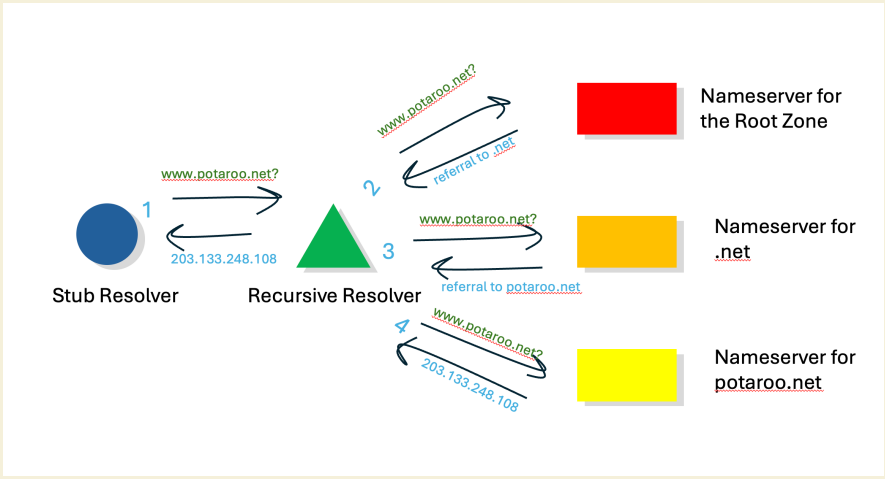

Figure 1 – Name Resolution in the DNS

Every DNS name is resolved in the same way. The name itself defines the order of name resolution processing, and defines the path to be followed through the distributed database that leads to the answer you are seeking.

In all this, there is one starting point for every DNS resolution operation: the root zone.

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

However, I think this view understates the criticality of the root zone in the DNS. The DNS is 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 in its provided configuration. 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 in a recursive manner.

If this was how the DNS actually worked, then it's pretty obvious that the entire DNS system would've melted down years ago. 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 must be queried.

However, behind this statement lurks an uncomfortable observation. If all of 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 cached entries time out, they would be discarded from these local resolvers (as they could not be refreshed by re-querying the root servers). The light of the DNS would slowly fade to black bit by bit as these cached entries time out and are removed. The DNS root zone is the master lookup for every other zone. That's why it deserves particular attention. For that reason, the DNS root zone is uniquely different from every other zone.

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 in resolvers. With a relatively small root zone and a relatively small set of DNS recursive resolvers, then the root zone query load should be small. Even as the Internet expands its user base the query load at the root servers does not necessarily rise in direct proportion. It's the number of DNS resolvers that supposedly determines root server query load if we believe in this model of the root's function in the DNS.

However, the model does not appear to hold up under operational experience. The total volume of queries per day recorded by the Root servers since January 2016 is shown in Figure 2.

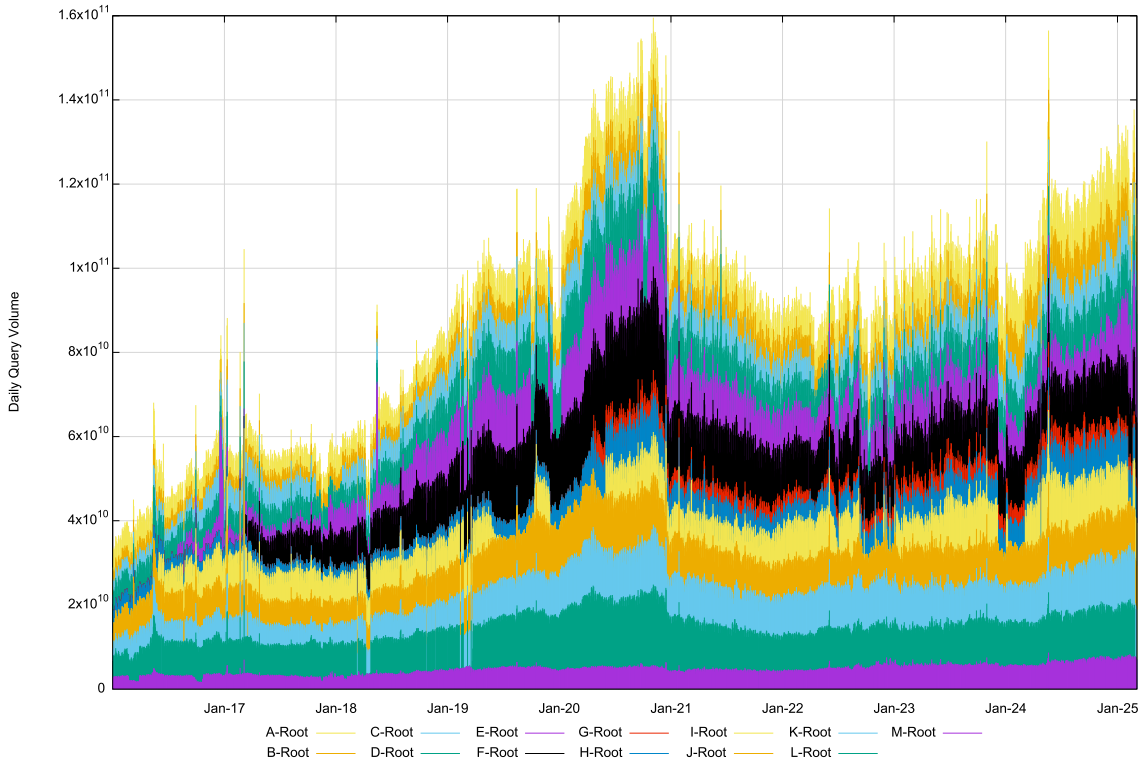

Figure 2 – Root Service Queries per day – from https://github.com/rssac-caucus/RSSAC002-data

Over the period from 2016 to 2020 the volume of queries seen by the collection of root servers had tripled. The query volume came down in 2021 and stabilised over 2022. It is likely that changes to the behaviour of the Chrome browser may explain this abrupt change. Chrome used to probe the local DNS environment by making a sequence of queries to non-existent names (so-called *Chromeoids*) upon startup and because the query names referred to undelegated top-level domains, these queries were a significant component of the queries seen at the root servers. Changing this behaviour in Chrome at the end of 2020 appears to have resulted in a dramatic change to the DNS query profile as seen by the root servers. However, over 2023 and 2024 the aggregate volume of queries seen by the root servers has resumed its upward trend, rising by 40% from some 90B queries per day at the start of 2023 to more than 130B queries per day at the start of 2025.

What are we doing in response to this trend in the growth of queries to the root zone? How are we ensuring that the root zone service can continue to grow in capacity in response to this resumption in the growth of query rates?

Digression – The Economics of the DNS

In conventional markets, when a good is consumed, the consumer pays the producer a fee for the consumption of that good. As long as the fee covers the cost of production of the good, then increasing consumption generates increasing revenue what can cover the costs associated with expanding the means of production of the good. Obviously, that's a very simplistic view of the operation of markets, but the key assumption is that greater consumption generates more revenue for producers, which, in turn, allows producers to produce greater volumes of the good. The essential assumption is that there is an underlying market-based discipline associated with the production and consumption of the good.

This assumption breaks down in the DNS, and in the root zone servers in particular. DNS queries are essentially unfunded. Like many Internet users I have an Internet Service Provider (ISP), and I pay an access fee for their service. Typically, an ISP operates a DNS recursive resolver for its clients, and my access fee contributes to my ISP's costs in running this resolver service. However, it's a fixed access fee, not a metered fee, so I contribute the same sum to the running of this shared resolver when I submit one DNS query per day or one million!

As well as the costs in operating this resolver service does the ISP incur any other cost in operating a DNS service to resolve my queries? No! All of the authoritative nameservers that are queried by my ISP's resolver are not funded by my ISP. More generally, all DNS queries in the public Internet are not directly funded by the querier!

Obviously, there are costs associated with operation of authoritative nameservers, and, for the most part, these costs are met by the "owners" of the zones which are served by these nameservers. There are various funding models for authoritative nameservers, ranging from metered costs per answered query, flat rate costs and even free services in some circumstances. But the essential aspect of this service is that authoritative nameservers do not derive revenue from the entities that query them. If there is a revenue stream it comes from the DNS zone administrators who are paying for the nameservers to serve their zone.

I did note that this holds "for the most part" and there is one very notable exception here, namely the root zone. The twelve entities who provide the nameservers for the root zone do so as a collection of independent largely autonomous volunteers who meet their own costs.

This is in many ways a curious relic of an earlier Internet that had a spirit of cooperative enterprise in many of its endeavours, but at the scale where each Root Service Operator is operating a service platform capable of responding to an average query load of some 10B queries per day, then this is no slight donation of effort and resources to a common good outcome. Such a core of altruism in the centre of a market-driven frenzy of activity that operates today's digital world is unusual to see.

Given the criticality of the role that these operators collectively undertake, and the observation that directly or indirectly we are all beholden to the outcomes of these efforts to maintain a functional namespace for the Internet then perhaps, odd as it may be, this is a better situation than many of the alternatives.

In a market economy, a monopoly supplier of a critical resource is able to extract a monopoly rental from all others, while customers cannot seek relief through competitive offerings because of the very nature of the monopoly. Today's world looks to market regulators and the associated public regulatory frameworks to protect markets from such forms of abuse. But in the Root Service function we find a service that is both universal across the entire collection of individual public regimes, and a collective monopoly. A self-imposition by these operators of a freely offered service is perhaps not the only possible

response to counter such risks of potential abuse of role, but so far, the ethos of these twelve root service operators has proved to be an adequate and sufficient measure.

But perhaps it's now time to take the outstanding question, namely "How are we ensuring that the root zone service can continue to grow in capacity in response to this resumption in the growth of query rates?", and now factor in the apparent need to escalate the level of resources that are in effect donated to this service by this small collection of operators.

Root Zone Scaling

The original model of authoritative servers in the DNS was based on the concept of unicast routing. A server name had a single IP address, and this single server was located at a single point in the network. Augmenting server capacity entailed using a larger server and adding network capacity. However, such a model does not address the issues of a single point of vulnerability, nor does it provide an optimal service for distant clients.

More Servers

The DNS approach to this is to use multiple name server records. A DNS resolver was expected to retry its query with a different server if its original query did not elicit a response. That way, a collection of servers could provide a framework of mutual backup. To address the concept of optimal choice, DNS resolvers were expected to maintain a record of the query/response delay for each of the root servers and prefer to direct the majority of their queries to the fastest server.

Why not use multiple address records for a single common server name? The two approaches (multiple server names and multiple address records for a name) look similar. Once a resolver has assembled a collection of IP addresses that represent the nameservers for a domain, then it seems to me that a resolver could be justified for treating the list of IP addresses consistently, irrespective of whether the list was assembled from multiple IP addresses associated with a single name, or from multiple names. The use of multiple names allows for the use of multiple paths through the DNS to resolve these names of the nameservers can remove a potential single point of failure, although I wonder as to the true benefit of using a set of nameserver names within a common single DNS zone as compared to using a single name with multiple IPv4 and IPv4 Resource Records, particularly when the bulk of DNS zones are provisioned with 2 or 4 nameservers, so there are typically 2 or 4 IPv4 and IPv6 addresses. I suspect that the use of multiple names is a policy compliance outcome rather than a true effort to provision nameservers with resilience through diversity.

If we want to increase the capacity of the Root zone, then why not just add more nameserver names to the root zone? What's so special about this zone's use of 13 named nameservers and a total of 26 IP addresses? For the Root zone, the scaling issue with multiple nameservers is the question of completeness and the size of the name server response to the *priming query*. The question here is: If a resolver asks for the nameservers of the root zone, should the resolver necessarily be informed of all such servers in the response? The size of the response will increase with the number of servers, and the size of the response may exceed the default maximal DNS over UDP payload size of 512 bytes.

The choice of the number of server names for the root zone, 13, was based on the calculation that this was the largest list of a server list that could fit into a DNS response that was under 512 bytes in size. This assumed that only the IPv4 address records were being used in the response. With the addition of the IPv6 AAAA records the response size has expanded. The size of the priming response for the root zone with 13 dual stack authoritative servers is 823 bytes, or 1,097 bytes if the DNSSEC signature is included, and slightly larger if DNS cookies are added.

In today's DNS environment, if the query does not include an EDNS(0) indication that they can accept a DNS response over UDP larger than 512 bytes, then the root servers will provide a partial response in any case, usually listing all 13 names, but truncating the list of addresses of these services in the Additional Section of the response to fit with a 512 byte payload.

There have been experiments in the past with more than 13 nameservers servers at the apex of a DNS-like name system (such as the Yeti project, of some 5 - 8 years ago) and while it is technically feasible to do so, some vexed questions remain, such as how to select new root service operators, what is a safe ceiling of the number of such services, how would it impact the stability and coherence of the name system.

Until we have much broader levels of adoption of query name minimisation than we appear to have today, then root servers are privy to the myriad of domain names that users are querying, which is effectively a real-time view into the activity in the Internet through this meta-data query stream. If we opened up the root service to a broader set of operators, would be temptation to monetise this unique and highly valuable data stream prove overwhelming? In this space is it even possible to enforce constraints that would preclude any such activity?

So far, we appear to have avoided such difficult questions by leaving the number of root nameservers constant and scaling the root service in other ways.

If we can't, or don't want, to just keep on adding more root servers to name server set in the root zone, then what are the other scaling options for serving the root zone?

More Service Platforms

The first set of responses to these scaling issues was in building root servers that have greater network capacity and greater processing throughput. But with just 13 servers to work with, then this was never going to scale at the pace of the Internet. We needed something more.

The next scaling step was 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). Now the routing system is used to optimise the choice of the "closest" location for each root server.

Root	Sites
A	59
B	6
C	13
D	220
E	328
F	359
G	6
H	12
I	85
J	148
K	131
L	123
M	23
Total	1,513

Table 1 – Anycast Site Counts for Root Servers, March 2025 (https://root-servers.org/)

The Root server system has embraced anycast, some more enthusiastically than others. There are a currently a total of 1,513 sites where there are one or more instances of root servers. Some 24 months earlier, in January 2023, the root server site count was 1,396, so that's a 8% increase in the number of sites in a little over two years.

The number of authoritative server instances 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 at each site. Today, there are a total of 1,907 instances of root server systems.

Even this form of expanding the distributed service may not be enough in the longer term. We are seeing the resumption of the growth profile last seen in 2016-2020. With a 25% compound annual query growth rate in four years from now we may need double the root service capacity from the current levels, and in a further four years we'll need to double it again. Exponential growth is a very harsh master.

Can this anycast model of replicated root servers expand indefinitely? Or should we look elsewhere for scaling solutions?

Query Deflection for Negative Responses

There have been many studies of the root service and the behaviour of the DNS over the past few decades. If the root servers were simply meant to respond to the cache misses of DNS resolvers, then whatever is happening at the root is not entirely consistent with such a model of behaviour. 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 50% of root zone queries result in NXDOMAIN responses (Figure 3). The NXDOMAIN response rate was as high as 75% in 2020, and dropped presumably when the default behaviour of the Chrome browser in using *Chromeoids* changed. In theory these queries are all cache misses at the recursive resolver level, so the issue is that the DNS is not all that effective in handling cases where the name itself does not exist.

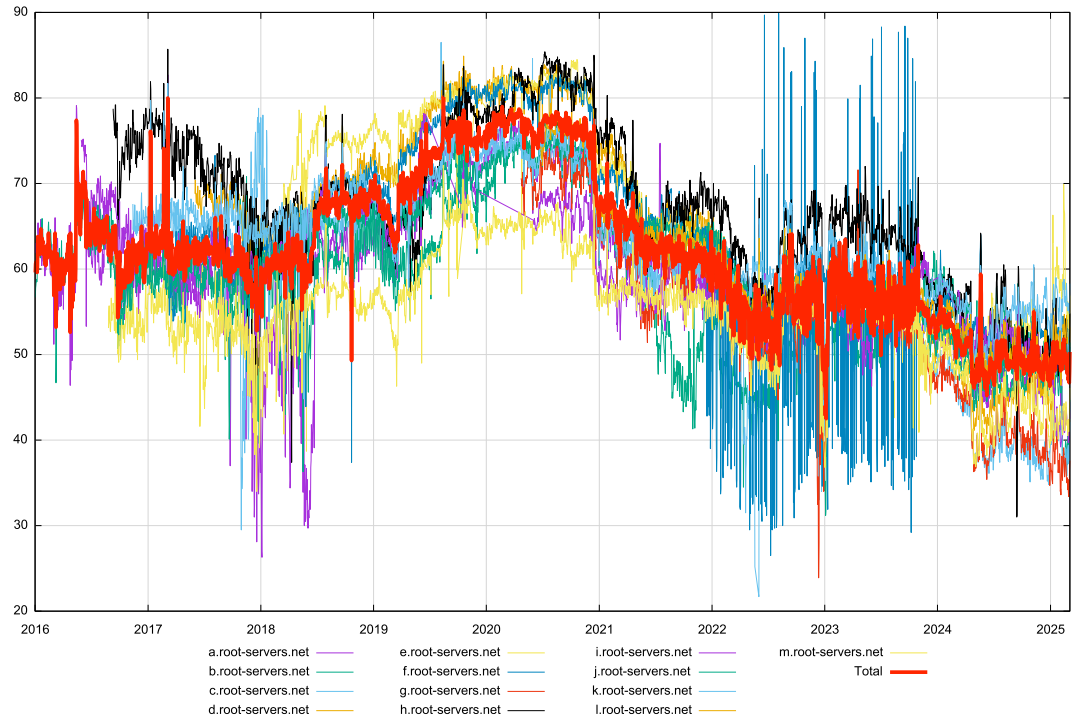

Figure 3 – Proportion of Root Zone NXDOMAIN responses per Day (RSSAC002 data)

If we want to reduce the query pressure on the root servers then one possible approach is to alter the way DNS resolvers handle queries for non-existent names, and in particular names where top-level label in the queried name is not delegated in the root zone. How else can we deflect these queries away from the root server system?

One such approach is described in RFC 8198, Aggressive NSEC Caching. When a top-level label does not exist in a DNSSEC-signed zone, and the query has the EDNS(0) DNSSEC OK flag enabled, the NXDOMAIN response from a root server includes a signed NSEC record that gives the two labels that exist in the root zone that "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, then 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" cached records.

If a recursive resolver cached both the 1,443 top level delegated labels and the 1,444 NSEC records in the root zone, then the resolver would not need to pass any queries to a root server for the lifetime of the cached entries. If all recursive resolvers performed this form of NSEC caching of the root zone, then the query volumes seen at the root from recursive resolvers would fall significantly for non-existent labels.

How many TLDS are in the Root Zone?

There are 1,443 top level domains in the root zone of the DNS in March 2025. It has not always been this size. The root zone started with a small set of generic labels, and in the late 1980's expanded to include the set of two-letter country codes. There were some tentative steps to augment the number of generic top level domain names, and then in the 2010's ICANN embarked on a larger program of generic TLD expansion. Figure 4 shows the daily count of TLDs in the root zone since 2014.

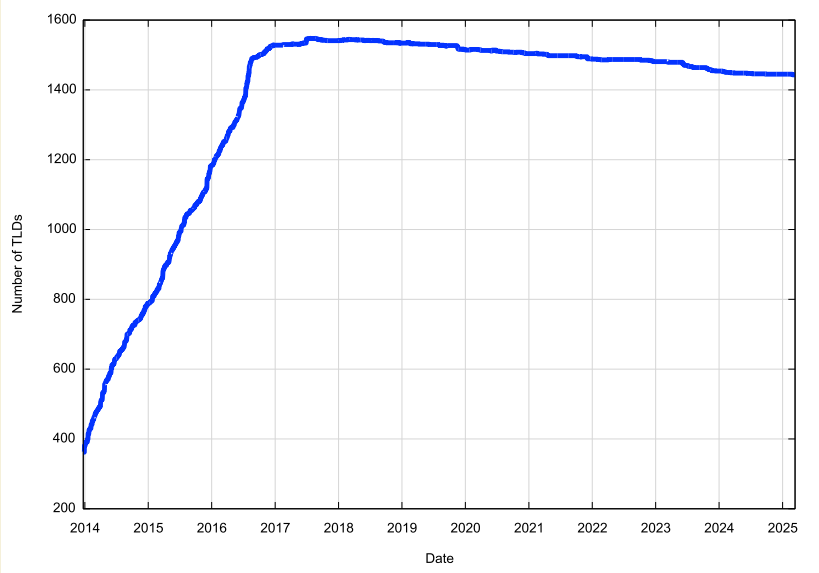

Figure 4 - Daily count of root zone TLDs

What was surprising to me was that top level domains are not necessarily permanent. The largest TLD count occurred in August 2017 with 1,547 TLDs, and since then the number of TLDs have been declining since.

Aggressive use of NSEC caching in recursive resolvers appears to play a contributory role in helping us scale the root zone. Bind supports this function as of release 9.12. Unbound supports this as of release 1.7.0. Knot resolver supports this as of version 2.0.0. But the queries at the root zone keep growing despite the declining proportion of queries resulting in an NXDOMAIN response. While this measure may have dampened the relative growth of queries for non-existent names seen at the root servers, to some extent, it has not had a major impact on the overall issue in the growth of queries directed to the root servers, as there are other factors that appear to be driving this.

I'd characterise the situation as aggressive NSEC caching representing a tactical response to root zone scaling concerns, as distinct from a strategic response. The technique is 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.

Root Zone Mirroring

Another scaling option is to jump completely out of the query/response model where recursive resolvers incrementally learn the contents of the root zone query-by-query and simply load the entire root zone into their local cache and refresh this local copy with a period of several hours or even a day or so. The idea here 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 here the existence of the LocalRoot service that apparently offers DNS NOTIFY messages when the root zone changes. The root zone is not a big data set. A signed, uncompressed plain text copy of the root zone as of 14 March 2025 is 2.2 Mbytes in size.

However, this approach has its potential drawbacks. How do you know that the zone you might have received via some form of zone transfer or otherwise is the current genuine root zone? Yes, the zone is signed, but not every element in the zone is signed (NS records for delegated zones are unsigned). The client is left with the task of performing a validation of every digital signature in the zone, and at present there are some 1,444 RRSIG records in the root zone. Even then the client cannot confirm that its local copy of the root zone is complete and authentic, because of the unsigned NS delegation records in the root zone.

The IETF published RFC 8976, the specification of a message digest record for DNS zones, in February 2021. This RFC defines the ZONEMD record.

What's a Message Digest?

A *message digest* is a form a condensed digital signature of a digital artefact. If the digital artefact has changed in any way, then the digest will necessarily change in value as well. If a receiver of this artefact is given the data object and its digest value then the receiver can be assured, to some extent, that the contents of the file have been unaltered since the digest was generated.

These digital signatures are typically generated using a *cryptographic hash function*. These functions have several useful properties. They are normally a fixed length output function, so that the resulting value is a fixed size, irrespective of the size of the data for which the hash has been generated.

They are a *unidirectional* function, in that knowledge of the hash function value will not provide any assistance in trying to recreate the original data. They are *deterministic*, in that the same hash function applied to the same data will always product the same hash value. Any form of change to the data should generate a different hash value. Hash functions do not necessarily produce a unique value for each possible data collection, but it should be exhaustively challenging (unfeasible) to synthesise or discover a data set that produces a given hash value (*preimage resistance*),

and equally challenging to find or generate two different data sets that have the same hash function value (*collision resistance*).

This means that an adversary, malicious or otherwise, cannot replace or modify the data set without changing its digest value. Thus, if two data sets have the same digest, one can be relatively confident that they are identical. Second pre-image resistance prevents an attacker from crafting a data set with the same hash as a document the attacker cannot control. Collision resistance prevents an attacker from creating two distinct documents with the same hash.

The root zone includes a ZONEMD record, signed with the Zone Signing Key of the root zone. When a client receives the root zone it should look for this record, validate the RRSIG of the ZONEMD record in the same way that it DNSSEC-validates any other RRSIG entry in the root zone, then use the value of this record and compare it with a locally calculated message digest value of the local copy of the root zone. If the digest values match, then the client has a high level of assurance that this is an authentic copy of the root zone and has not been altered in any way.

The dates in the DNSSEC signatures can indicate some level of currency of the data, but further assurance at a finer level of granularity than the built-in key validity dates that the local copy of the root zone data is indeed the current value of the root zone is a little more challenging in this context. DNSSEC does not provide any explicit concept of revocation of prior versions of data, so all 'snapshots' of the root zone within the DNSSEC key validity times are equally valid for a client.

The root zone uses a two week signature validity period (Figure 5).

```
. 86400 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2025031303 1800 900 604800 86400
. 86400 IN RRSIG SOA 8 0 86400 20250326200000 20250313190000 26470 . nYhmvV[…]Ng==
```

Figure 5 – Root Zone SOA signature

This approach of a whole-of-zone signature has some real utility in terms of the distribution of the root zone to DNS resolvers and thereby reduce the dependency on the continuous availability and responsiveness of the root zone servers. The use of the ZONEMD record allows any client to use a local copy of the root zone irrespective on the way in which the zone file was obtained. Within the limits of the authenticated currency of the zone file as already noted, any party can redistribute a copy of the root zone and clients of such a redistributed zone can answer queries using this data with some level of confidence that the responses so generated are authentic. It would be useful to augment the existing in-band root zone retrieval using AXFR with a simple memorable web retrieval object, such as https://1.2.3.4/root_zone.txt, for example, to allow the zone distribution function to be undertaken by CDNs as well as by DNS servers.

For those resolvers who elect to use a locally managed copy of the root zone, the ZONEMD record can be used to verify the authenticity of a received root zone. Resolver implementations that perform this verification using ZONEMD include Unbound (from v1.13.23) and PowerDNS Recursor (from v4.7.04) and Bind (v9.17.13).

Notification mechanisms that could prompt a resolver to work from a new copy of the root zone are not addressed in this ZONEMD framework. To me that's the last piece of the framework that could promote every recursive resolver into a peer root server. We've tried a number of approaches to scalable distribution mechanisms over the years. There is the structured *push* mechanism where clients sign up to a distributor and the distributor pushes updated copies of the data to clients. Routing protocols use this mechanism. There is the *pull* approach where the client probes its feed point to see if the data has changed and pulls a new copy if it has changed. This mechanism has some scaling issues in that aggressive probing

by clients may overwhelm the distributor. We've also seen hybrid approaches where a change indication signal is pushed to the client, and it is up to the client to determine when to pull the new data.

This model of local root zone distribution has the potential to change the nature of the DNS root service, unlike NSEC caching. If there is one thing that we've learned to do astonishingly well in recent times its distribution of content. Indeed, we've concentrated 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 digital object, then the CDN infrastructure is perfectly capable of feeding this zone to the entire collection of recursive resolvers with ease. 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 could eliminate the entire notification superstructure. Each iteration of the root zone contents would be published 2 hours in advance and is valid for a period of precisely 48 hours, for example. At that point the root zone can be served by the existing millions of recursive resolvers rather than the twelve operators and some 2,000 server instances we use today. That's a thousand-fold increase in the capacity of the root system, which at the same time eliminating the general reliance on narrow neck of incremental queries being directed to the 12 root server operators that underpin today's DNS.

Futures

We operate the root service in its current framework because it represents a set of compromises that have been functionally adequate so far. That is to say the predominate query-based approach to root zone distribution hasn't visibly collapsed in a screaming heap of broken DNS yet! And it will probably continue to operate in a robust manner for many years to come.

But we don't have to continue relying on this query-based approach just because it hasn't broken so far. Our need to further scale this function is an ongoing need, and it makes a whole lot of sense to take a broader view of available options and the just-in-time delivery process used by DNS's incremental query name resolution algorithm.

We have some choices as to how the root service can evolve and scale.

With Aggressive NSEC Caching we can have recursive resolvers make better use of signed NSEC records we appear to have staved 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 DNS system to collapse and then try and salvage the DNS from the broken mess, or perhaps we could explore some alternatives now, and look at how we can break out of a query-based incremental root content promulgation model and view the root zone as just another content "blob" 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 volumes of queries to ever more demanding clients, and perhaps also provide an alternate answer to the continual questions about the politics and finances relating to root servers and their operation.

The reason why content distribution networks have revolutionised the Internet in recent years is that pre-provisioning at the edge makes for a faster, cheaper and more scalable network in the current context of abundant computing and storage capabilities. If we are prepared to allow this same thinking to intrude into the way we provision the DNS, then I suspect there are similar benefits that could be realised for the DNS as well.

Disclaimer

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

Author

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

www.potaroo.net