December 2014

Geoff Huston

Workshop on DNS Future Root Service

The theme of a workshop, held at the start of December 2014 in Hong Kong, was looking at means to enable further scaling of the root server system, and the 1½ day workshop was scoped in the form of consideration of alternative approaches to that of the default of adding further anycast instances of the existing 13 root server anycast constellations. There were two specific proposals that were considered at this workshop that formed the workshop's focus.

This was a workshop operating on at least three levels. Firstly there was the overt agenda of working through a number of proposed approaches that could improve the services provided by the DNS root service. The second was an unspoken agenda concerned with protecting the DNS from potential national measures that would "fragment" the DNS name space into a number of spaces, which includes, but by no means not limited to, the DNS blocking activities that occur at national levels. The third level, and an even less acknowledged agenda, is that there are various groups who want to claim a seat at the Root Server table.

These various agendas work against each other to some extent, in that increasing the number of root servers runs the elevated risks of a lack of managed control of the contents of the root, which exacerbates the risk of fragmentation of the root zone contents. Introducing the capability for any resolver to serve the root zone allows for the root service to grow atomically on a needs and capability basis, which addresses the perception of imposed scarcity of root services in certain parts of the world, but at the same time undermines the privileged role of the current root service operators, some of whom have created sustaining business models of leasing out anycast instances of their root service, while others who have aspirations to run a root server would like to see the this privileged role maintained. So its not surprising to see some conservative reactions from some of the incumbents when faced with proposals that radically change the root service environment. From this respect this was always going to be a workshop that aired ideas and discussed concepts, as distinct from a workshop that focused on viable solutions or attempted to reach any form of closure on any particular proposal

The highly positive aspect of this two day event was bringing the Chinese, and in particular BII, ZDNS and CNNIC, into the same room as some folk from the DNS root server community and the IETF DNS technical group. And in this respect I would say that the workshop was a success - it exposed the Chinese to the considerations of the DNS root server community, and at the same time the Chinese had the opportunity to expose some of their work and thoughts about expanding the root service.

Aside: The Root Server System

The DNS is a hierarchically distributed naming system. A name in the DNS is a sequence of labels that read from left to right, where is a sequence of labels can be views in a pairwise fashion where the label on the left is the "child" of the "parent label on the right. The common progenitor of this name structure is the root, which is notionally

defined as the trailing "." at the right most part of a fully qualified DNS name.

The DNS system is implemented as a collection of authoritative name servers. A set of name servers is nominated as being authoritative for a given zone, and any of these name servers will provide identical answers in response to queries for names that lie within its zone. When queried for a name that is a child of this zone these servers will respond with the authoritative name servers for that child zone. The root zone is a zone file that contains a zone header and a list of delegations of the complete collection of top level names.

When a recursive resolver is passed a query relating to a name about which it has absolutely no knowledge it will generate a query to a root server. The response is not the desired information, but the name servers that are authoritative for the top level domain name being queried. The recursive resolver will then query one of these name servers for the name, and will receive in response the name servers that are authoritative for the send level domain name, and so on. The inference from this mode of operation of the DNS is that the operation of the DNS requires that all resolvers have access to at least one root name server at all times.

A single root name server is not a robust approach to root server design. One the other hand, millions of distinct root servers would lead to a host of other issues, not the least which include the considerations of root priming queries and the challenge of keeping a very large collection of distinct servers in tight synchronization with respect to the content that they serve as the root zone of the DNS. The adoption of 13 distinct root servers is a compromise between these two pressures. The exact number was the outcome of the number of distinct root server labels, and their IPv4 addresses that can be loaded into an unsigned IPv4 UDP response to a root server priming query that is less than 512 bytes.

The demands of scaling the root service in the face of an expanding network involved the adoption of anycast cloud servers for a number of the root name servers. The anycast structure has a number of major attributes that help the root server system. The first is that the multiple instances of the root server instance split up the query load against that server's IP address into the localities served by each anycast instance. This allows the root service instance to distribute its load, which improves its service. Equally, it allows the root server instance to appear to be "close" to many disparate parts of the client base simultaneously, which also contributes to an improvement in its service profile. This technique also allows the root server to cope with various forms of denial of service attacks. Wide scale distributed attacks are spread across multiple server instances, which implies that a greater server capacity is deployed to absorb the attack. Point attacks are pinned against a single server instance, which minimizes the collateral damage to a single instance of the anycast server set. It also imposes a two level hierarchy on distribution of changes to the root zone, limiting the number of points of direct notification of changes to the primary authoritative instance of the root zone.

The Proposals

Operating a root slave service on 127.x.y.z

This approach is not an architectural change to the DNS (or at least not intentionally). For recursive resolvers that implement this approach this is a form of change in query behavior in so far as a recursive resolver so configured will no longer query the root servers for queries it would normally direct to an instance of the root, but instead direct these queries to a local instance of a slave server that is listening on the recursive resolver's loopback address. This slave server is serving a locally held instance of the root zone, and the recursive resolver would perform DNSSEC validation of responses from this local slave to ensure the integrity of responses received in this manner. For users of this recursive resolver there is no apparent change to the DNS or to their local configurations. Obviously, there is no change to the root zone either.

The motivation behind this proposal is that there are a population of recursive resolvers that are still too far away from all of the root servers, and this causes delays in the DNS resolution function. The caching properties of recursive DNS resolvers is such that the overall majority of queries directed to the root servers are for non-existent top level domains, so a pragmatic restatement of the problem space is that there are recursive resolvers that take too long to generate a NXDOMAIN response, and this approach would reduce this time delay to a single query interval from the client to the recursive resolver if the resolver was coupled to a slave server that was serving the root zone.

However, given this particular formulation of the problem space, then the larger and more comprehensive the anycast constellations of the root servers, the less the demand for this particular approach. Locales where there are adequately close DNS root services from the anycast root servers would find no particular advantage in operating a local slave DNS root server as the marginal speed differential may not be an adequate offset for the added complexity of configuration and operation of the local slave server.

The linking of the root zone information to the loopback is a point of fragility in the setup. To set up a slave DNS server that is authoritative for the root zone it would uses multiple root servers to ensure that it has access to a root zone from at least one of the anycast server constellations at any time. If at any time it cannot retrieve a master then it should SERVFAIL, and the local recursive resolver should revert to conventional queries against the root servers.

This proposal provides integrity in the local root server through the mechanism of having the recursive resolver perform DNSSEC validation against the responses received from the local root slave. If the recursive resolver is configured as a DNSSEC-validating resolver then this is configurable on current implementations of DNS recursive resolvers. However, if it is desired to limit DNSSEC validation to just the responses received from the local slave root server then this is not within the current capabilities of the more widely used DNS resolver implementations today.

The advantages of this approach is that the decision to set up a local slave root server is a decision that is entirely local to the recursive resolver, and the impacts of this decision affects only the clients of this recursive resolver. No coordination with the root server operators is required, nor any explicit notification. The local slave server is only indirectly visible to the clients of this recursive resolver and no other.

"Unowned" Anycast Root Server Constellations

One possible motivation behind this proposal lies in the observation that before the root was DNSSEC-signed we placed much reliance in the concept that the root zone was served from only a

small number of IP addresses (the 13 root servers) and no other, but when the root zone is DNSSEC signed then the integrity of the responses generated from a root zone is based on the ability of the receiver of the response to validate the signed responses using their local copy of the root keys. Who serves the zone in such a context is largely irrelevant.

This approach uses the same root key set to sign a second root zone, changing the NS records, and also maintaining a second root hints file corresponding to the new addresses behind these new root servers. However, this second set of addresses would not be exclusively assigned to further set of root server operators, but allowed to be used by any party, in a form of uncontrolled anycast.

In some ways this proposal is similar to the existing AS112 work, where anyone can set us a resolver to respond to queries for commonly queries non-existent top level domains (such as .local) with NXDOMAIN responses. The implication is that if there is a perception that a locale is poorly served by the existing root server anycast constellations then a local instance of this anycast root server can be set up and served from an instance of this unowned anycast address. As the address is specifically dedicated for unowned anycast there is no need to coordinate either with the existing root server operators or with other operators of root zone servers on the same anycast address. A prospective operator of one of these root servers simply serves the unowned anycast root zone from one of the small pool (2 address couplets, each linking an IPv4 and a IPv6 address) of reserved anycast addresses. Recursive resolvers would query this server by using a distinct unowned anycast root hints file.

Why would any recursive resolver trust the veracity of any responses received from one of these unowned anycast root servers? One could well ask the same of recursive resolvers who query into any of the 13 anycast constellations for the root zone, and to some extent the risks of being led astray are similar. The one mitigation of the latter case is that to hijack an existing anycast constellation prefix requires the coercive corruption of the routing system to inject a false instance of an "owned" address, while there is no such concept as hijacking of the unowned anycast prefix. However, in both cases some skepticism on the part of the recursive resolver is to be encouraged, and recursive resolvers should be motivated to validate all such responses using the local copy of the DNS trust anchor material. This is still a part of "good housekeeping" recommended operational practice for recursive resolvers using the existing root servers, but is a more strongly worded requirement for resolvers using this unowned anycast service.

Commentary

While it could be regarded as a byproduct of a single hierarchical name space, the centralization of root information in the DNS is operationally problematical and does not cleanly fit within a distributed and decentralized peer model of a network architecture. The adoption of root server anycast constellations is an attempt to respond to this, to an extent, by over-provisioning of this critical service. A similar picture is emerging in the area of content provisioning, where cloud-based content is essentially an exercise in over-provisioning, where single points of distribution are replaced by a larger scale of multiple delivery points in order to improve the quality of the delivered service.

However, end users don't enjoy the same level of control, and are dependent on external conditions that are effectively out of their direct control. Not only does this result in highly variable service experiences, but it also leaves the user highly exposed to various forms of online surveillance. The distributing computing world has created external dependencies for users where access to local service is reliant on external availabilities. The DNS is a good example of this scenario, in so far as resolution of a DNS name that is not already contained in local caches requires priming queries to external servers. Obviously these dependencies on external services is an incredibly fragile system where local services cannot be reliably provided using only local infrastructure.

Secondly, the exposure of full name queries to servers high in the hierarchy exposes excessive amounts of information about users, the applications and names that they use and the locales of the services that

they use. Tapping the DNS query stream from a root is a high payback activity if you want to glean intelligence about what users do. This is not just about global names, but also extends to names used exclusively in a local context (e.g. myprinter.myplease.example.com),where part of the cache priming query set leaks to external authoritative DNS servers.

Perhaps it need not be explicitly stated, but neither proposal contemplates any splitting of the name space itself, nor any change to the role of the IANA as the administrative entity responsible for the contents of the root zone, nor any changes to the existing root name servers. However, both proposals essentially unlock the root zone and allow others to serve the zone contents without any formal coordination with the existing root servers, the IANA or any other server who may also be serving the zone.

It is also interesting to note that neither proposal advocates opening up more distinct anycast root server constellations (i.e. adding a 14th root server). It is completely unclear what additional technical value would be gained by adding any new server constellations to the existing collection, as distinct from further expansion of the existing anycast server constellations. The related question about the politics of the 13 root server operators and the politics of augmenting this group in various ways is an entirely different subject, and was not considered at this workshop in any manner.

Nor do these proposals advocate any form of alteration of the control of the anycast server instances of these existing root server systems. In both cases they propose to create a completely novel structure which is distinct from the existing structure of recursive resolvers, the root zone (and hints) file and the set of root servers.

Both proposals are incremental in nature, and propose a form of augmentation to the existing structure of recursive resolvers and the root name server, rather than any change to the existing structure. But in so doing there is a distinct possibility that this form of uncoordinated piecemeal expansion at a local level could prove to be more effective across the Internet, and the critical role of the 13 root server operators would diminish over time if this were the case.

Neither approach is not entirely without some forms of change. All these uncoordinated root server operators would mean that root zone changes via NOTIFY is not feasible, so it would be back to periodic zone transfers and timers in the root zone headers. There is no longer a quick mistake correction capability in the root zone if served in this way, although it could be argued that the massive level of caching of DNS information actually implied that any changes in the root zone were subject to cache flushing in any case, irrespective of the speed of zone change at the level of the root server anycast constellations.

What is perhaps more worrisome is that the unowned anycast proposal is in effect a proposal to fork the root one, and recursive resolvers are forced to position themselves within one regime or the other. The only common glue left in this environment is the root key, as the only way that a client of either regime can detect that they are receiving genuine answers is to perform response validation using the root key. This is placing a massive amount of invested trust in a security artifact that is only used today by a very small subset of recursive resolvers.

It also needs to be noted that our experience to date with owned anycast has been very poor. At one stage the IPv6 transition experimented with a form of unowned anycast in the form of 6to4 tunnel servers, and the results were hardly reassuring. Anycast clouds follow routing, not geography, and diagnosing operational failures that occur within an uncoordinated anycast structure can range from the merely challenging to highly cryptic and insoluble. The relationship between a recursive resolver and the actual root server it is querying is then occluded, and instances of structural failure in DNS name resolution are far harder to diagnose and correct. Considering that the name translation function is an essential foundation for the Internet, adding operational opacity to the root zone query function is not a step that should be taken lightly.

But that does not imply that the other proposal is free from operational concerns either. The complexity of the local slave resolver, with two concurrent DNS resolvers operating within the same host should also be questioned. While there is a current convention in DNS resolver and server deployment to avoid the model of a "mixed" mode resolver which is both an authoritative (or slave) server for some domains and a recursive resolver for all other domains, this is perhaps nothing more than a convention, and it seems overkill for a resolver to phrase a root query and reach through the loopback interface to ask a co-resident DNS resolver the queries that are being posed to the root. Why not just operate the local resolver in mixed mode and allow the root query to simply become a memory lookup within a single DNS resolver instance? Perhaps the only justification in this case is the issue of root zone integrity. In the mixed mode of a single resolver instance the question arises of how can the local resolver validate the contents of the transferred root zone is a reasonable question. In the loopback model, the local resolver performs DNSSEC validation of the root zone responses and therefore does not necessarily need to separately validate the contents of the transferred root zone. In theory a mixed mode resolver could DNSSEC validate the responses retrieved from its local instance of a slave zone server before passing them to the recursive resolver function, but its not clear than any existing DNS resolver implementations perform this form of DNSSEC validation of internal queries in a mixed mode of operation. Alternate approaches of including a zone signature to ensure integrity are also a possibility to ensure that the recursive resolver is not placed into a position of inadvertently serving corrupt root zone data.

It's evident that this approach of a local loopback slave resolver, and related variants, are already in use in the Internet. Some DNSSEC-validating recursive resolvers interpret the NSEC responses in the root zone as authentic denial of existence of TLDs as evidenced by the NSEC RRs, and are then able to generate an NXDOMAIN without asking the root directly. (There is some compelling evidence from the Day-In-The-Life (DITL) root server query data that the Google PDNS servers are performing some form of NXDOMAIN generation without referral to root servers, either through a local slave or through NSEC inference., and the component of queries from Google PDNS servers appears to be disproportionately small in the DITL root server query logs)

There were a few more observations about the state of the DNS and the role of the root server system to the larger DNS environment. Its true that we've placed considerable change pressure on the DNS in recent years. We've added IPv6 as both a DNS query transport protocol and as a query target in the form of AAAA records, and we've added DNSSEC to allow a client to validate the response that they receive from a resolver if they so desire. We've added encoded character sets (rather than attempt enforce an 8-bit "clean" DNS) to support non-ascii character sets in DNS labels, and we are now expanding the root zone with more top level label names. We've now placed much reliance in dynamic DNS with incremental updates and incremental change notification from the authoritative server to its secondary slave servers. We've added extensions through EDNS0 flags, and then started to populate that extension with various options and flags that alter the protocol itself (DNS UDP response size), the content of responses (DNSSEC flags) and even the particular response (experimenting with the Client Subnet option). We are now toying with information suppression to avoid the gratuitous verbosity of DNS, restricting the query fields passed to servers to the minimal level of information necessary to provide a useful response through query name minimization. Today's DNS is not your grandparent's DNS, nor even your older sibling's DNS! It's changing.

So to what extent are we bound by overarching considerations of keeping archaic forms of DNS functional, as against making changes that either break older DNS resolvers or expose them to higher levels of vulnerability? We are seeing this in the two proposals considered in this workshop where in both cases the resolvers need to perform DNSSEC validation on root zone responses in order to ensure that they are not being misled by rogue resolvers masquerading as authentic slave root servers. Older resolvers that do no use DNSSEC validation are not welcome in this particular playground. But if this is the case about particular proposals related to expanding the availability and robustness of the root zone, then why not question the entire nature of the root zone priming query itself?

Why are there 13 root nameservers? Because, so goes the story, the response to a priming query made to a root server fits within the defined minimum accepted UDP payload limit in the DNS of 512 octets, as defined in that most venerable of specifications, RFC1035 ("Domain Implementation and Specification,", P. Mockapetris, November 1987). Really? Well that's not longer the case. A root priming response these days includes IPv6 records for all bar the "e" root server instance, and the response is 755 octets. And if you add the DNSSEC-OK flag to the query, then the response is 913 octets in size. Yes, there are still resolvers that are not equipped with EDNS0, and its true that if you limit the response size to 512 octets then the IPv4 addresses of 13 of the root nameservers and the IPv6 addresses of two of the root nameservers squeeze into 512 octets, but should this somewhat archaic constraint really be the defining limitation of the entire DNS system?

Other options include use of TCP as the transport protocol for the root zone priming query, which would remove this size-related constraint on the number of root zone servers listed in the priming response. That line of argument then immediately opens up the next question of if you are not constrained to the magic number of 13, then is there a larger number that makes technical sense?

> One possible answer is that no single number makes more or less than any other, based on a well known observation from Computer Science that there are only three "natural" numbers, 0, 1 and n!

So at this stage one is left with the view that while the number 13 meets the limitations of some archaic recursive resolvers (that should perhaps be retired in any case), there aside from this consideration there is no particular benefit at a technical level why 13 anycast constellations of root name servers would be any better or worse than 14.

One possible approach is to allow any recursive resolver to be a slave server for the root zone, as described above. If the zone transfer function included an integrity check across the entire transferred zone (such as a hash of the transferred zone, signed by the root's zone signing key), then the recursive resolver could be assured that it was then serving an authentic copy of the root zone. However such an integrity check on the transferred zone only assists the local recursive resolver that it has obtained an authentic and current copy of the zone. Clients of that recursive resolver should be as skeptical as ever and ask for DNSSEC signatures, and perform DNSSEC validation over all signed responses that are received from the recursive resolver. The advantage of this approach is that it is an implementation of an unlimited number of root name servers, where every recursive server that so wants to be can serve its own authentic copy of the root zone. The disadvantage is that, like all large distributed systems, there is some introduced inertia into the system and updates take time to propagate. However, in a space that is already highly cached then difference between what happens today and what would happen in such a scenario may well be very hard to see.

Conclusions

As noted at the start this was a workshop that was not directed toward conclusions, but more to a peer review form of activity where a number of concepts around possible directions for the DNS were intensely thrashed for a couple of days. For the workshop itself there were no particular conclusions.

For myself I think I have two conclusions to carry forward.

The first is that DNSSEC is indeed an extremely valuable asset for the DNS, and we can move forward with a larger and more robust system if we can count on various efforts to subvert the operation DNS being thwarted by all forms of clients of the DNS, even to the level of applications performing name to number resolution, insisting that they are exposed to the DNS responses and their signatures, and

performing validation on the received data. For many resolvers its as simple as adding "dnssec-validation yes;" to your local DNS resolver configuration. Just do it, and do it now!

The second is more generic. We are finding it increasingly challenging to react to inexorable pressures of scaling the Internet's infrastructure while still maintaining basic backward compatibility with systems that only conform to technical standards of the 1980's. We need to move on. The 512 octet limit in the DNS is over. Resolvers should support EDNS0, or shut themselves down. Resolvers should perform DNSSEC validation, or shut themselves down. We need to move on and look at how to support a larger, more responsive, more robust and far less chatty name system, and we need to be able to contemplate heading away from basic binary encodings over unencrypted UDP packet exchanges if that's what's needed. Wondering why there are 13 available slots for root name server operators, and wondering about how we could alter their composition, or change the interaction with the DNS root to support one or two additional root servers are really a totally unproductive conversations at so many levels. It may well be time to unlock the mainstream DNS and release the binds to its historic legacy, and contemplate a different DNS that does not involve these arbitrary constraints over the number and composition of players.

The attraction of unconstrained systems is that local actors can respond to local needs, and respond by using local resources, without having to coordinate or cross-subsidize the activities of others. Much of the Internet's momentum is driven by this loosely constrained model of interaction. If we can use the possibilities opened up by a securing the DNS's payload, where who passed the DNS information to you is irrelevant, but whether the information is locally verifiable is critically important, then and only then can we contemplate exactly what it would take to operate an unconstrained DNS system.

Disclaimer

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