The ISP Column
A column on various things Internet

February 2024
Geoff Huston

DNS OARC 42

The DNS Operations, Analysis, and Research Center (DNS-OARC) brings together DNS service operators, DNS software implementors, and researchers together to share concerns, information and learn together about the operation and evolution of the DNS. They meet between two to three times a year in a workshops format. The most recent workshop was held in Charlotte, North Carolina in early February 2024. Here are my thoughts on the material that was presented and discussed at this workshop.

DELEG and Delegation

In brief, the proposed DELEG Resource Record (https://datatracker.ietf.org/doc/draft-dnsop-deleg/) is a parent-side record that combines the semantics of specifying a zone cut (delegation) and the names of the delegated zone's nameservers (NS records) with their IP addresses (Glue records) with the additional capability to add a DNS transport identity, port address and a relative priority associated with each listed name server.

The DELEG record can also allow the nameserver name to use alias records, which allows redirection of the resolver client to some other name target (a function that NS records cannot do, as, according to the DNS specifications, CNAMES cannot be used as the target of NS names). Unlike NS records, the DELEG record is authoritative in the parent zone, and can be signed by the parent zone's DNSSEC key, if the parent zone is DNSSEC-signed. DELEG is intended to sit side-by-side with NS and Glue records, so a resolver that is not explicitly aware of how to handle a DELEG record still has conventional delegation information in a referral response. My description of the DELEG record (and some personal opinions thrown in for free!) can be found at https://www.potaroo.net/ispcol/2024-02/deleg.html.

The "how" of doing this augmented form of DNS delegation is underway within the IETF standards process, and while there may be more in the way of IETF ruminations in the coming months, I would hazard the opinion that there is little that will substantively change from the basic structure already provided in the initial proposal.

While the broad mechanics of “how” have been well described, the “why”, or the nature of the motivations to augment one of the core elements of the DNS, is not so well understood, at least not by myself in any case!

Firstly, the copy of the name server (NS) records held by the parent, and passed to a client in a referral response are not authoritative for the parent. The delegated child zone is the authority point for these records. When this consideration is applied to NS records, then the child will DNSSEC-sign to NS records in the child zone, with the child zone’s key. The parent cannot sign its copy of these NS records with the parent zone’s key. The NS records in the parent are therefore unsigned. DELEG is proposed as a record for which the parent zone is authoritative, and the DELEG record is signed by the parent zone key. In this manner, such delegations now are signed parts of the parent zone. This signing of the delegation points is intended to counter attempts to attack this aspect of DNS resolution, by enabling a validating client to detect unauthorised efforts to substitute a redirection within the delegation function.

But does this ability to validate a delegation make any sense in the context of DNSSEC? DNSSEC is defined to allow a client to validate the currency and authenticity of a DNS response that they receive. It does not matter how that response was obtained. If DNSSEC validation fails, then the response should not be used by a client. If the response is not DNSSEC-signed, and the client is able to determine that the zone is unsigned, then the client cannot determine whether the response is authentic or not. Even if all the delegation points along the path from the root zone to the answer were DNSSEC signed in some manner, and each delegation point were to be validated (incurring some penalty in the performance of resolution), then an unsigned response from an unsigned end child zone is still subject to the potential for substitution attacks and the response is still not inherently trustable. If you really want such a level of assurance about the authenticity of DNS responses, then there really is no alternative to DNSSEC-signing of the response that is drawn from the end child zone. Signing the individual delegations in the top-down delegation path used to reach that response neither adds nor detracts from the outcome of validation of the response itself. Validation of DELEG records from a verifiability stance appears to add cost without corresponding benefit.

Secondly, if the objective is to allow the recursive resolver to query the authoritative server using an encrypted transport, then there are two questions that would need to be answered. In the general case, where is the privacy benefit in protecting this channel? The original end user is not identified by this recursive-to-authoritative query (RFC 7871, Client Subnet, is indeed a hideous privacy mistake!) so the privacy aspect is somewhat marginal in the general case. Secondly, the characteristics of the interaction between a recursive resolver and an individual authoritative server are almost the opposite of that of the stub resolver-to-recursive. In the latter case the overheads of setting up an encrypted channel can be amortised over the anticipated volume of subsequent queries between the two parties, and the privacy aspects of not exposing the stub resolver identity in the query stream are readily apparent. Neither is necessarily the case in the recursive-to-authoritative situation. The high setup cost is simply an added cost to the DNS resolution transaction without any readily apparent change in the privacy posture of such transactions.

Finally, the one aspect of this DELEG proposal which appears to have some attraction is the ability to perform an alias translation of nameserver names. NS names cannot be CNAMES, and this has been a source of some frustration in the DNS service environment. The ideal situation would be for a nameserver name in the parent to be the name if an alias target in the DNS operator's name space, which could be further aliased to a specific name (or names) to optimise the name resolution for the client, in a manner analogous to the use of CNAMES in content hosting. With aliases this can be achieved without any further reference to either the parent or child zone operators. This property is behind the enthusiastic adoption of CNAME records by the content hosting sector. While this may sound compelling as a rationale, the issue is how would the parent and child also publish NS records for backward compatibility with non-DELEG aware resolvers? if the desire of the DELEG alias is to allow the DNS some independence (and flexibility) as to how they map the nameserver name to an IP service destination, the requirement to maintain a concurrent legacy NS record which is not an alias would appear to frustrate this desire.

So, I am left wondering why is DELEG evidently so attractive to many DNS folk? What are their reasons for giving this proposal far more attention than their predecessors, such as

- the work on the REFER RR type (https://datatracker.ietf.org/doc/html/draft-jabley-dnsop-refer-00),
- ns-revalidation (https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-ns-revalidation-00), or
- the proposed delegation information signer (https://datatracker.ietf.org/doc/html/draft-fujiwara-dnsop-delegation-information-signer-00)?

Presentation: DELEGate the Modern Way by David Lawrence
https://indico.dns-oarc.net/event/48/contributions/1041/attachments/1006/1926/DNS-OARC 42 DELEGations++ (1).pdf

DNS Resilience

If there is one aspect of the DNS that stands out it's the way in which the DNS has incorporated resilience into its core design. The DNS resolution protocol was built upon simple stateless transactions using UDP as the preferred transport. This approach can be extremely efficient, cheap and fast. But UDP is not a reliable transport and the design of both the infrastructure and the protocol takes this inherent unreliability into account. A zone is served by multiple named nameservers. If one nameserver is unresponsive, a querier can always re-query using a different nameserver. It's just UDP, and stateless queries are simple and have little in the way of overhead. Therefore, zones are served by multiple named nameservers. This is also passed to the next level, as a nameserver may have multiple IP addresses, not only to incorporate dual stack functionality, but also to allow a logical nameserver to have multiple points of connection to the network. If one IP address is unresponsive to a query, then a client can always use another IP address.

Adding multiple nameservers does not always ensure improved resilience. The story of the Facebook DNS outage of October 2021 shows that even with multiple DNS servers, clustering them in a single fate-shared BGP routing state can still bring everything down (https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/). Then there was the DYN outage of October 2016 where a, single DNS provider, DYN, was overwhelmed by an exceptionally large denial of service attack (https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn).

Perhaps resilience is not just about multiple name servers with multiple addresses, but also using multiple providers with multiple points of presence in the routing system. In theory, multiple providers should avoid a single point of vulnerability and potential failure and allow client recursive resolvers to work around the failure of nameservers.

How well does this work?

Shane Kerr, from IBM's NS1 service, set up an experiment using an unsigned zone served by both NS1 and Amazon's Route 53. The zone contained a single wildcard short TTL TXT record, with a different response when served by each provider. The queries were made by RIPE Atlas using regular DNS queries.

The experiment was to take one provider's service offline and observe the time taken to resolve the DNS name before, during and after this service outage from one provider. The expectation is that with the onset of the outage a number of clients would experience a DNS timeout if they queried the now-unresponsive server, but they would still be able to resolve the name once they queried the servers from the other provider. During the outage the resolver clients should "latch" onto the available name servers and DNS resolution performance should return to the original state before the outage. The subsequent restoration of the service should not affect observed DNS resolution performance.

What was observed was that some resolver families (open DNS resolvers operated by a single provider, or recursive resolvers operated by a single ISP) behaved as described here, but for others there was no noticeable change in DNS resolver performance at all, while in others the slower resolution times were noted across the entire period of the outage. Such resolvers apparently were persistent in querying the unavailable server and did not cache the server's unavailability.

Multiple providers can certainly increase in overall availability of a service, but failure modes can still impair resolution service. These issues relate to the behaviour of the recursive resolvers rather than the authoritative servers, and the diversity of behaviours in this impaired service environment points to diverse code bases for recursive resolvers.

The DNS remains surprising. Using multiple DNS services for a DNS service can improve service resilience, but also introduces sone unanticipated instabilities.

Presentation: Using Multiple Authoritative Vendors May Not Work Like You Thought by Shane Kerr
https://indico.dns-oarc.net/event/48/contributions/1035/attachments/1001/1951/Using Multiple Authoritative Vendors Does Not Work.pdf

UDP, Truncation and TCP

I've been told by someone who was there at the time that the original design decision to base the DNS design on a maximum UDP payload of 512 octets corresponded to the size of a sector read from a disk drive. In any case, this maximum size of a DNS payload was a match to the largest IP packet that a host was assured to reassemble, namely 576 octets, once you take IP headers into account.

All this has changed since then, but not by much. Ethernet raised the defacto IP packet size to 1,500 octets (or thereabouts) and the IPv6 specification defined a minimum unfragmented packet size of 1,280 octets. We operate the network now with three somewhat informal bands of packet sizes: any packet of size 1,280 or smaller is highly likely to be passed through the network without requiring fragmentation, and any packet larger than 1,500 octets is equally likely to require fragmentation and any packet in between is in the grey zone any may require fragmentation.

The problem is that fragmentation is highly uncertain. A number of attacks rely on efforts to inject false fragments into a packet stream and because fragments have no transport header, firewalls are often in a quandary as to permit or deny them. Many simply deny them.

The DNS response to this situation is to make use of a maximum UDP buffer size, provided as a DNS extension (EDNS), where this size specifies the maximum payload of a DNS response that can be passed using UDP. When the response is larger than this size, then the response should be truncated and the truncation bit (TC bit) is set in the DNS header to indicate this incomplete state of the response (RFC 2181, Sec. 9). When the client receives such a packet it should discard the truncated UDP response and re-query using TCP.

Akamai uses `akadns.net` in its hosting configuration, and about a year ago decided to make some steps to improve its posture in supporting IPv6 by adding IPv6 glue records to its `akadns.net` referral response. This additional data lifted the size of the referral response for this domain from 344 octets up into the grey area of 1,454 octets.

At this time the `.net` servers were experiencing bursts of UDP queries that were almost double the baseline rate, and a burst of UDP responses with the TC bit set (Figure 1).

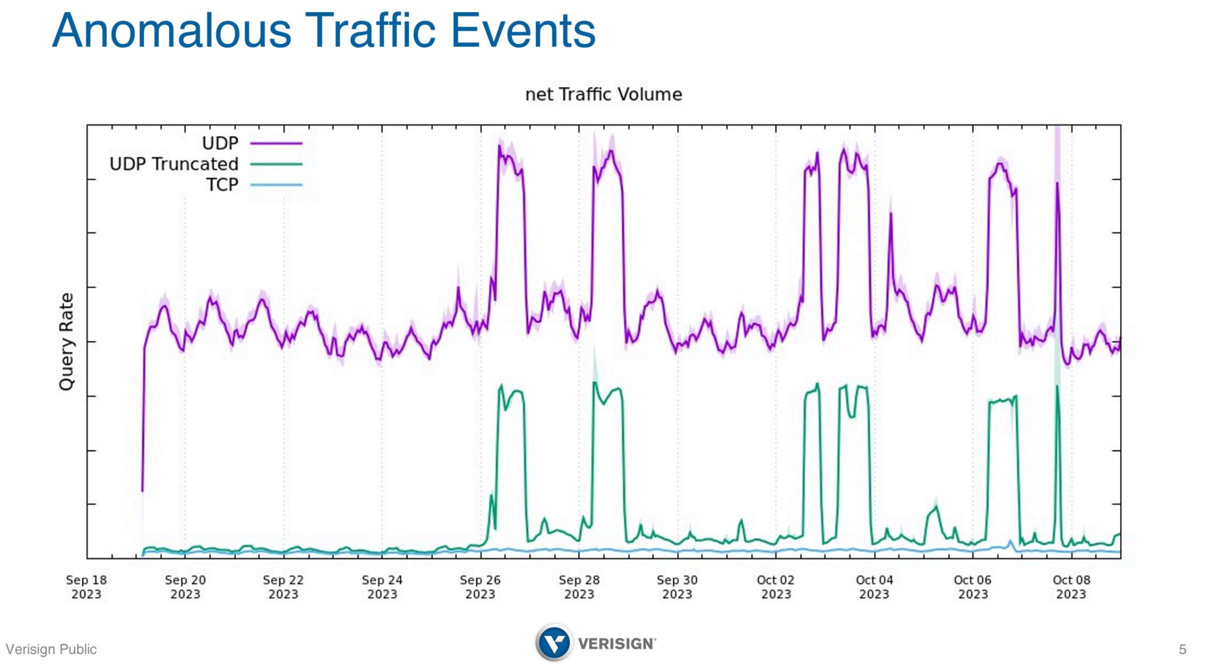


Figure 1 – Anomalous Traffic seen at .net, from Real World Challenges with Large Responses, Truncation, and TCP , Duane Wessels, Ralf Weber, Slide 5

What was anomalous here was that the TCP rate was not changing at all (the blue trace in Figure 1). This situation was resolved to a European ISP who was issuing TCP SYN packets to start a TCP session in response to the truncated UDP response, but not closing the TCP connection handshake by sending the final ACK of the 3-way TCP handshake. It was established that the ISP uses Linux *iptables* with connection tracking, which sometimes become full. The result of these full iptables was that TCP SYN packets were permitted outbound, but either the returning SYN+ACK was rejected, or the internal TCP state could not complete, so no handshake's closing ACK was sent. This failure to start a TCP session caused the resolver to retry the query over UDP at an accelerated rate.

Between the ISP, Akamai or the `.net` operator, Versign any one of these parties could've resolved this situation. The ISP could've increased the size of its *iptables*, Akamai could've resolved this by reducing the number of IPv6 glue records in its referral response, or Verisign by relaxing its implementation of the requirements for including all of the glue records in its referral response. Its implementation of referral responses was strictly aligned to the advice in RFC 9471:

> "This document clarifies that when a name server generates a referral response, it MUST include all available glue records for in-domain name servers in the additional section or MUST set TC=1 if constrained by message size."
> RFC 9471

Verisign tested a less strict glue truncation policy on a single site (the anycast site closest to the ISP) during a spike event, and aggressive query traffic dropped to zero across all Verisign `.net` servers within two seconds of making the change. Akamai removed some glue records from the referral response which immediately dropped the UDP truncation rate.

A year later, and Akamai are using a single IPv6 glue record in their referral response.

```
$ dig +norecurse akadns.net @a.gtld-servers.net
; <<>> DiG 9.18.21 <<>> akadns.net @a.gtld-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40597
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 10, ADDITIONAL: 7
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;akadns.net.                  IN      A

;; AUTHORITY SECTION:
akadns.net.           172800  IN      NS      a3-129.akadns.net.
akadns.net.           172800  IN      NS      a7-131.akadns.net.
akadns.net.           172800  IN      NS      a11-129.akadns.net.
akadns.net.           172800  IN      NS      a1-128.akadns.net.
akadns.net.           172800  IN      NS      a9-128.akadns.net.
akadns.net.           172800  IN      NS      a5-130.akagtm.org.
akadns.net.           172800  IN      NS      a28-129.akagtm.org.
akadns.net.           172800  IN      NS      a13-130.akagtm.org.
akadns.net.           172800  IN      NS      a18-128.akagtm.org.
akadns.net.           172800  IN      NS      a12-131.akagtm.org.

;; ADDITIONAL SECTION:
a3-129.akadns.net.    172800  IN      A       96.7.49.129
a7-131.akadns.net.    172800  IN      A       23.61.199.131
a11-129.akadns.net.   172800  IN      A       84.53.139.129
a1-128.akadns.net.    172800  IN      A       193.108.88.128
a1-128.akadns.net.    172800  IN      AAAA    2600:1403:11::80
a9-128.akadns.net.    172800  IN      A       184.85.248.128

;; Query time: 23 msec
;; SERVER: 192.5.6.30#53(a.gtld-servers.net) (UDP)
;; WHEN: Fri Feb 09 06:28:14 EST 2024
;; MSG SIZE  rcvd: 372
```

I must admit that I am still scratching my head over Akamai's generous use of nameservers here. While one nameserver is too few, ten is probably too many. No resolver client is going to work through all ten nameservers before giving up, so the only conclusion is that this is some crude form of DNS-based load balancing across the set of nameservers, or some enduring artefact of historical legacy thinking that "If thirteen was a good number for the root zone then its good enough for me?". A more conventional approach these days would be to use a smaller set of name server names and use anycast to distribute the load across their network of servers.

It's also useful to question the assumptions behind the strict requirement of RFC 9471 when the set of glue records starts to bloat. If a referral is causing UDP message truncation then things have gone badly wrong in the first place with the design of the zone's nameservers, but to complicate this by forcing the parent to serve the referral response by TCP is adding to the fragility of the situation, rather than increasing the resilience of the delegation.

Presentation: Real world challenges with large responses, truncation, and TCP, Duane Wessel and Ralf Weber
https://indico.dns-oarc.net/event/48/contributions/1036/attachments/1004/1920/challenges-truncation-tcp-combined v2.pdf

Caching of DNS Resolution Failures

There is a tension between servers and clients in the DNS. Clients are interested in obtaining a positive response as quickly as possible, while servers would prefer clients to accept the response that they've been given and if the answer is not definitively positive or negative, then avoid thrashing about to find out if some other query combination might elicit a definitive answer. The issue arises with DNS responses that are neither positive (this is the answer to your query) nor negative (there is no such domain or no such resource record). These indeterminate responses, such as SERVFAIL, REFUSED, timeouts, and FORMERR, often cause the resolver client to seek an alternative name server, an alternative query formulation, or just try again to see if the previous condition was transitory.

If a stub resolver has multiple recursive resolvers, then such indeterminate errors will generally trigger the stub resolver to re-query using the other configured recursive resolvers. If an authoritative server has multiple name servers, then the recursive resolver may re-query using each of the name servers. It's easy to see that the result is a potential combinatorial query burst. If a stub resolver is configured to use 3 recursive resolvers and the name in question has 10 name servers (see the above example for `akadns.net`), and each name server has both an IPv4 and an IPv6 address, then an initial indeterminate response may lead to a further 59 queries.

RFC 9520 has some further advice on this behaviour, namely that "A resolver MUST NOT retry a given query to a server address over a given DNS transport more than twice (i.e., three queries in total) before considering the server address unresponsive over that DNS transport for that query." It's not clear to me that this advice is enough for the quite common case of a potential combinatorial explosion. In our example of a potential 59 followup queries, no recursive resolver has re-queried a single nameserver address over the same DNS transport. So, each individual recursive resolver may be following this guidance, yet the query amplification may still occur.

It appears that the admonition about moderating repeat queries in RFC 9250 is reasonable, but perhaps it misses a crucial point. When considering the number of name servers to use for a domain, perhaps some advice about trying to avoid "too much of a good thing" is also appropriate. It seems unlikely that recursive resolvers will temper their search for an answer in the face of indeterminate error responses, so reducing the scope of a combinatorial explosion of choices of resolvers may well be helpful.

How many name servers (and name server IP addresses) is the "right" number to balance the demands of operational resilience and timeliness with the desire to avoid combinatorial explosion of failure-induced followup queries?

What bounds on repeat query pacing strikes a workable balance between rapid recovery when encountering an isolated failure condition and avoiding a query race condition that imposes exceptional load states on authoritative server infrastructure?

There are both open questions, in so far as we see all kinds of answers in the DNS environment, but little in the way of a shared understanding in the DNS world of a convergence to set of commonly shared answers.

In many ways this is also an aspect of the failure of the economics of the DNS. Queries are free to the querier but come at an incremental cost to the responder. Queriers optimise their behaviour by performing repeat queries when encountering unexpected conditions, as this happens without cost to the querier. It is quite predictable to see responding servers appeal for restraint on the part of queriers, but equally predictable to see such appeals to be largely ignored by the querying resolvers!

Presentation: The Impact of Negative Caching and DNS Resolution Failures by Yannis Labrou
https://indico.dns-oarc.net/event/48/contributions/1045/attachments/996/1947/DNS-OARC42_Labrou_Thomas_v1_OARC42.pptx

DNSSEC Upgrades

There are not that many choice points when you DNSSEC-sign your zone. You need to pick an encryption algorithm and related key size, and you need to select wether the negative record structure is NSEC, Compact NSEC, or NSEC3. Yet even in these choices there is scope for variation and some novel situations to encounter.

What if you were administering the `.gov` signed zone, signed with RSA and using NSEC3 records, and you wanted to shift your zone across to a different DNS operator who used ECDSA P-256 with NSEC compact denial of existence and an online signer? Conventional wisdom would say: "Drop DNSSEC, then shift the NS records to the new operator and then re-sign the zone". But what if you wanted to maintain the zone to be DNSSEC-signed across the transition?

The approach taken when shifting `.gov` from Verisign (RSA/NSEC3, pre-signed zone) to Cloudflare (ECDSA/Compact NSEC, front end signer), was to use an RSA front-end signer in Cloudflare and shift the zone. This led to a zone that variously had NSEC3 and Compact NSEC records, depending on which server provided a signed response (which is an option in RFC 8901, Sec. 5.2). This then allowed the next step of having the zone to be exclusively served by the incoming zone operator. The next step is to transition the signing algorithm from RSA to ESDSA, which is well traversed territory using a period of dual signing with both algorithms before retiring the legacy algorithm from the zone.

We might have had experience in DNSSEC-signing algorithm upgrades, but there are two aspects that still instil some justified caution. There is rolling the algorithm at the root zone of the DNS and rolling the algorithm in extremely large domains. The latter has been undertaken by Verisign, who reported on their experience in shifting from RSA to ECDSA for the `.com`, `.net` and `.edu` zones. Verisign followed the conventional double signer approach (RFC 6781, sec. 4.1.4), where the key with new algorithm was activated and the entries in the zone were progressively double-signed with the new and old zone signing keys. At this point the new keys can be added to the zone's DNSKEY record and the DS record added to the root zone. The RSA keys can then be removed, and the RSA-generated signatures removed from the zone and the transition is complete.

The issue where is adding additional signatures to signed responses and additional keys to the DNSKEY record inflates the sizes of these records. Here we re-enter the world of large DNS response, UDP, truncation and TCP re-queries. The "worst case" in this situation are NXDOMAIN responses, which have four signatures, and all such responses were exceed 1500 bytes during the rollover. We would expect these large DNS responses to be truncated during this phase of the algorithm transition and the TCP query rate to increase at a corresponding rate. This not exactly what occurred.

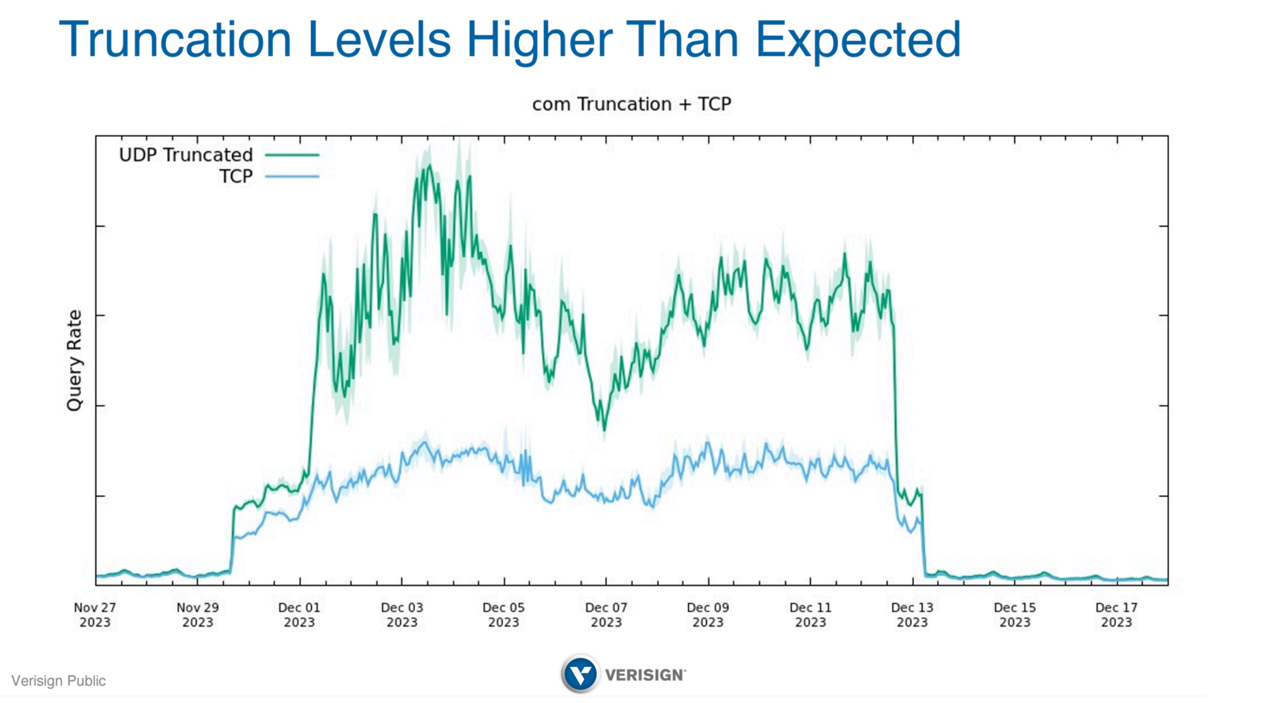


Figure 2 – Truncation Levels during DNSSEC Algorithm Transition, from Verisign's Transition to ECDSA, Duane Wessels, slide 20.

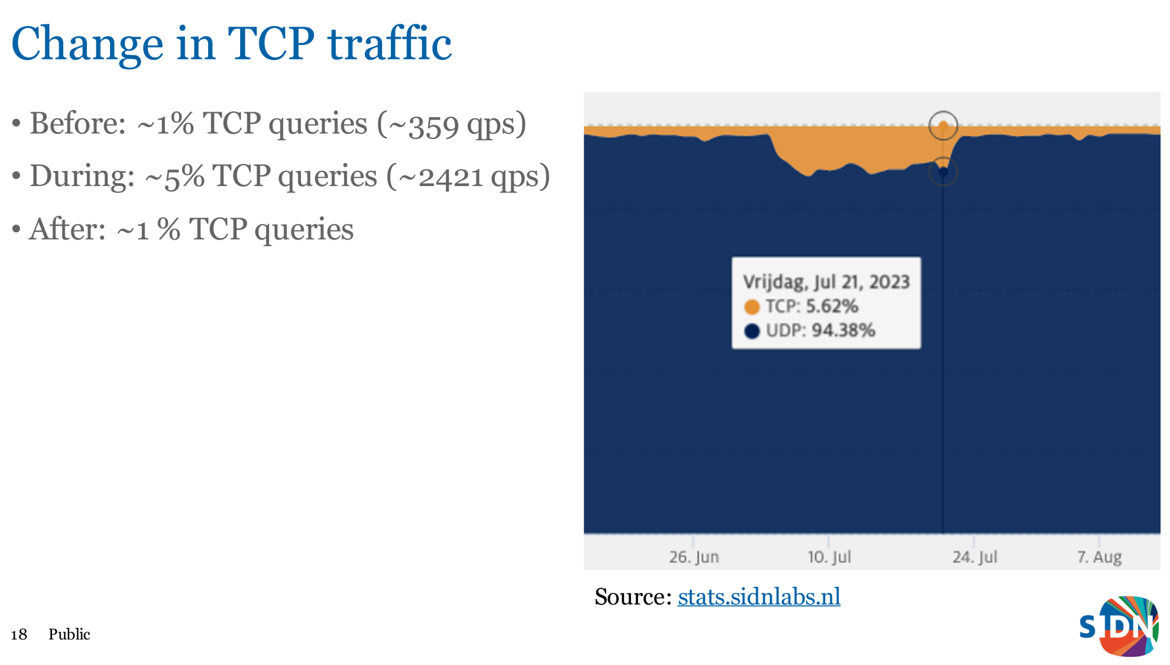
Prior to the transition there was a small rate of truncated UDP and a similar rate of TCP connections, presumably relating to queries that were using a very low UDP buffer size of 512 bytes or similar. During the initial phase of the transition, when a subset of delegation entries in the zone were double-signed, the UDP truncation rate increased, but the TCP connection rate lagged behind at some 60% - 70% of the truncated UDP rate. The truncated UDP rate rose to a second level in the second day of this progressive double-signing phase, but curiously the TCP connection rate lagged even further. At one point the TCP connection rate was some 30% of the truncated UDP rate. The same intermediate state occured at the end of the transition (Figure 2).

This data tends to suggest that there are a significant number of resolvers, or possibly a smaller number of resolvers with a significant number of clients, that cannot perform a fallback to TCP when presented with a truncated UDP response, and an associated supposition that this situation also triggers such resolvers to perform a more aggressive search for alternative servers by repeating the UDP query, with the same combinatorial query amplification factor that we have already noted.

This may have resulted in some level of query loss, but it was query loss that related to DNS names that did not exist in the first place (NXDOMAIN), so it is not clear that such reports about non-answers for non-names would ever have been lodged with the zone operator in the first place!

This data points to an on-going issue with the DNS' ability to handle large responses. The current conventional approach for the DNS these days is to avoid the pitfalls of fragmentation by performing response truncation at relatively conservatively chosen sizes (such as at 1,232 bytes of payload) and rely on the truncation signal to direct the resolver to re-query using TCP. This transition data points to a sizeable population of users who are behind recursive resolvers that do not behave in this way.

The .NL top level domain performed a similar algorithm transition in July 2023, also from RSA to ECDSA. Similarly to the Verisign experience, the largest temporary response size was for NXDOMAIN responses while the zone was dual signed, rising to 1,402 bytes in size. Again, they observed an increase in TCP queries in the transition, presumably strongly influenced by generating signed NXDOMAIN responses and the interaction with the UDP buffer sizes used by both the clients of the service and the authoritative servers. The presentation gave no data about the rate of truncated UDP responses, and therefore there was no indication of the "conversion rate" from truncated UDP to TCP in this case.



Change in TCP traffic

- Before: ~1% TCP queries (~359 qps)
- During: ~5% TCP queries (~2421 qps)
- After: ~1 % TCP queries

Vrijdag, Jul 21, 2023
● TCP: 5.62%
● UDP: 94.38%
26. Jun
10. Jul
24. Jul
7. Aug

Source: stats.sidnlabs.nl

18 Public

Figure 3 – Change in TCP traffic ruing Algorithm Roll from KSK Algorithm Roll for .nl, Stephan Ubbink, slide 18

There were evidently no operational reports of outages in this transition, but as we have already observed, if the issues were concentrated in the lack of a received response for queries about names that did not exist in the first place, then such a situation may well go unreported!

Presentations:

GOV multi-signer transition with NSEC/NSEC3 by Christian Elmerot
https://indico.dns-oarc.net/event/48/contributions/1038/attachments/1005/1948/gov-transition-nsec-nsec3.pdf

Verisign's Transition to ECDSA by Duane Wessells
https://indico.dns-oarc.net/event/48/contributions/1043/attachments/1003/1927/wessels-verisign-algrolls.pdf

KSK algorithm rollover for .nl by Stefan Ubbink
https://indico.dns-oarc.net/event/48/contributions/1044/attachments/990/1942/dns-oarc42_SIDN_KSK_algorithm_rollover_for_.nl.pdf

DNS over IPv6

Large DNS responses (by "large" I mean any value greater than 1,500 bytes) in the DNS are "tricky". Large responses in IPv6 are very much a lost cause. The failure rate of lost responses when a large response is passed from an IPv6-only authoritative server is now running at some 51% of individual transactions, which is clearly an unacceptable position. If anyone has dreams of an IPv6-only platform for the DNS, or even a more modest objective of supporting a dual stack DNS, then we clearly need to change this picture.

How can we improve the situation? The first part of the response is to avoid fragmentation completely. Both clients and servers should limit the response size in UDP to avoid fragmentation and should limit the MSS size in TCP to a similar setting. But this exposes a second issue. How many resolvers will open a TCP session in response to a truncated UDP response? Or to phrase this a slightly different way, what proportion of users are behind resolvers that are incapable of performing a followup with TCP when presented with a truncated response over UDP? We need to ensure that this proportion of TCP-challenged resolvers and servers is zero!

Presentation: Is the DNS ready for IPv6 by Geoff Huston
https://indico.dns-oarc.net/event/48/contributions/1047/attachments/993/1904/2024-02-01-dns-oarc.pdf

DNS Server Fingerprinting

In the vein of everything old is new again there is another DNS server fingerprinting tool being developed, this time by a group at the Université Grenoble Alpes. There was some work over a decade ago on this topic, using the difference between the response of various DNS software versions to identify the DNS server. As well as a paper published by a South Korean research group, there was the popular (at the time) tool by Miek Gieben, fpdns.

This latest effort works on the same principles as the previous tools. Ask some queries and then match the responses against the given templates and the tool may be able to identify the DNS server.

Presentation: DNS Fingerprinting by Yevheniya Nosyk
https://indico.dns-oarc.net/event/48/contributions/1042/attachments/994/1945/presentation.pdf

Cache Poisoning Attacks

It was almost 30 years ago when there was a successful cache poisoning attack that involved queries to Alternic and the use of bogus resource records in the query response. The basic response at the time was to tighten the rules of what was acceptable information in a referral response and discarding all other data. I understand that it also led to the DNS adopting the rather quaintly termed concept of "balliwick". The memory of the attack may well have faded but the term "in-balliwick" is still a thing in DNS circles!

Some 10 years later, the second such attack was the Kaminsky cache poisoning attack, where an off-path attacker attempted to guess the DNS query identifier and inject a response to the querier faster than the ordinary DNS. This crafted response contained incorrect glue records, and these were loaded into the resolver's cache. The response to this attack was to randomise the querier's source port, changing the number space for the attacker to successfully guess from 65,000 to some 4 billion.

A mere five years later we saw a fragmentation attack, enabled by the attacker being able to guess the IP ID value. The response was to randomise the querier's IP ID value, reduce the dependence on packet fragmentation in the DNS and use case randomisation in query names.

The exploration of these vulnerabilities continued. There was SADDNS, MaginotDNS and now there's TuDoor, each with some escalating complexity in terms of the steps the attacker needs to perform but in each case exposing some form of predictable behaviour on the part of the DNS resolver that exposes them to crediting a synthetic response as authentic.

I can't help thinking that the entire rationale of DNSSEC was that if everyone signed their DNS zones and if every resolver validated the responses they obtained, then this entire guessing game about how to pass off false data as authentic would be far more challenging.

On the other hand, misdirection in the DNS need not be catastrophic in any case. If the application uses the WebPKI to authenticate that the destination has an authentic claim to represent itself as the named service, then the misdirection will result in failed authentication in any case. I suspect that it's this reasoning that leads many service operators not to get overly concerned over these various obscure permutations of attacks on the DNS, even if their service name is not DNSSEC-signed.

Presentation: TuDoor Attack: Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets by Qifan Zhang
https://indico.dns-oarc.net/event/48/contributions/1039/attachments/992/1903/oarc42-tudoor-li.pdf

Detecting Vulnerabilities in DNS implementations

The DNS is a deceptive protocol in that it looks a lot simpler than it turns out to be. The operation of the protocol is described in 100's of RFCs and used ubiquitously by applications from user level apps through to totally autonomous service environments. There are now many implementations of various components of the DNS, and the name ecosystem now contains millions of resolvers, name servers and forwarders. To expect all these implementations to behave flawlessly is completely unrealistic.

So how do we detect these cases when the implementations stray into errant behaviour modes? Most of this seems to be trial and error, and where cases are detected, and sometimes exploited, CVEs are generated, fixes applied, and the cycles repeats itself. Endlessly.

An interesting response to this is the use of "fuzzing", where implementations are systematically checked by using test cases that explore query combinations where the range of values in the input fields is generated by fuzzing logic, and the divergence in responses between the expected and the actual response is detected.

The research group at UCI fuzz-tested six popular resolver implementations in various modes for recursion, forwarding and serving. This exercise was able to expose a number of vulnerabilities.

For decades the software industry has been able to amass some collective immunity from the consequent liabilities arising from the use of compromised or defective software. Maybe it's something to do with the densely written disclaimers that come with most software these days. Maybe it's also related to the observation that the more complex the task performed by the software, the more likely the case that it will encounter unanticipated situations and behave unpredictably. However, in my view that should not mean that they don't need to conduct extensive testing and active maintenance of their code in any case. It's great that such fuzzing tests exposed some issues with commonly used DNS software. It would be even better if the providers of these tools conducted such fuzzing tests themselves, as an integral part of the Q&A processes before each software release.

Presentation: ResolverFuzz: Automated Discovery of DNS Resolver Vulnerabilities with Query-Response Fuzzing by by Qifan Zhang

https://indico.dns-oarc.net/event/48/contributions/1037/attachments/1000/1913/ResolverFuzz-OARC42-zhang.pdf

NXDOMAIN

Surprisingly, much of the DNS infrastructure, particularly at the root of the DNS, is devoted to shovelling through dross. Some 60% to 70% of queries that are processed by the root zone servers result in NXDOMAIN responses.

Interestingly, recursive DNS resolvers see a different picture, In a recent (June 2023) study of queries presented to Cloudflare's Open DNS resolver, some 7% of queries refer to non-existent top level domains. This was a study of the queries, not a study of the responses, and there is a second category of queries that ask names within a delegated top level domain, yet still do not exist.

A study of such responses from a Chinese recursive resolver reveals that some 80% of the NXDOMAIN responses from the recursive resolver are generated from valid top level domain names.

The list of invalid top level domain names from this resolver is comparable to other studies, and includes the usual suspects of `.local`, `.localdomain`, and `.lan`. One of the commonly used cases of NXDOMAIN is of the form of a two-label string with an end user's UUID and `.local` appended. This appears to be related to WebRTC and video applications. The Chinese study found also two labels in high use that appear to be more local in scope, namely `.ctc` and `.cnp`.

What I'm still puzzled about is why the root zone attracts such a high level of such NXDOMAIN queries. The DNS would be a far better place if such queries were stopped at the recursive resolver. All recursive resolvers really should use the "hyperlocal" configuration as set forth in RFC 8806. But after four years of waiting for the folk running recursive resolvers to read and implement this measure, perhaps now it's time to reverse this situation. All configuration settings for the vendors' recursive resolvers should have hyperlocal for the root zone turned on as shipped in the default recursive resolver setting, and it should take an explicit local operator configuration change to turn it off!

Presentation: Analysis of NXDOMAIN data from an open resolver perspective in China by Jinghua Bai
https://indico.dns-oarc.net/event/48/contributions/1049/attachments/995/1934/dns-oarc42_DeepDive into NXDOMAIN Data in China.pdf

Performance Considerations for DNS Platforms

From my own experience in an operational context, when our service's DNS resolver platforms were dropping queries, then my first reaction was to try and increase the processor capability, add more memory and add more worker platforms into the resolver configuration. It's a brute force response, and perhaps some further subtlety in turning the platforms would've make a significant impact on the performance at much lower incremental cost.

Similarly, if your task is to undertake performance benchmark tests on various combinations of hardware platforms and DNS server implementations, then some specialised advice as to how to get the most out of the configuration will help. I won't got into details of the advice given by ISC's Petr Špaček and NS1's Jan Včelák, but it's clear to me that both presentations on performance benchmarking and tuning are well worth studying, and any summary I might provide here would not do them justice. Also, ISC's Ray Bellis has also authored a very useful fool to display the queues on your NIC card. It's at https://www.isc.org/blogs/ethq-linux-nic-monitoring-tool/.

Presentations:
DNS Benchmarking 101: Essentials and Common Pitfalls by Petr Špaček
https://indico.dns-oarc.net/event/48/contributions/1033/attachments/991/1943/pspacek.pdf

DNS Server Performance Tuning in Linux by Jan Včelák
https://indico.dns-oarc.net/event/48/contributions/1050/attachments/1007/1936/dns_server_performance_tuning_on_linux_draft.pdf

The DNS in 2024

The DNS OARC workshops have consistently managed to provide a good snapshot of the current topics of common interest that are driving the evolution of the DNS.

There is a push to load more service rendezvous information into the DNS, replacing a tedious and often inefficient probe exchange to discover the capabilities of service access points and match that with the capabilities of each client. Of course, pushing more information into the DNS is great, but trusting the authenticity of responses when a client attempts to pull it the data out again remains an unresolved issue. "Just use DNSSEC and sign everything" is a possible answer, but DFNSSEC has many moving parts and operators continue slip up on the delicate dance that is key management, and the result is all too often a case of self-harm when the validation outcomes from DNSSEC fail.

The larger response sizes that are a result of adding these security credentials to responses exacerbate one of the enduring weaknesses of the DNS protocol. While a UDP transaction is fast, efficient and very lightweight, this picture quickly darkens when the size of the payload enters the area of enforced packet fragmentation, packet loss and the overheads of a reliable stream-based protocol, encrypted or not. This would be tolerable if large packets were the rare exception, but when they become commonplace there is the added burden of context switching, head of line blocking potential and stream process state maintenance that add to the server load. There is some consensus that this is sustainable at the edges in the conversation between endpoint stub resolvers and recursive resolvers, but the picture is far murkier when looking at the inner conversation between recursive resolvers and the authoritative servers. We have a general consensus that our DNS wish list contains further improvements in speed, efficiency, flexibility, verifiability, and resistance to subversive attacks, but when it comes down to making choices about priorities and tradeoffs as to where to place the burden of incremental workload (and cost) any signs of consensus about next steps quickly dissipate.

As we scale up the DNS it is obvious that we are surviving not thriving, and the DNS living on a knife edge when trying to maintain an acceptable universal service and avoiding various outbreaks of local catastrophe. Part of the issue we face today is that the DNS appears to be imbalanced. There appears to be far more capability to pass queries inward to the DNS infrastructure than there is the capacity to answer all these queries. The presentation from Akamai and Verisign over the implications of TCP failure are slightly worrisome in that a single ISP's resolver infrastructure can push the server capacity into a less than ideal position. Do recursive resolvers need to be more restrained in their re-query behaviour when encountering server failure? Is the UDP-influenced philosophy, namely that when encountering service resolution issues, the best response is to increase the sending rate of repeat queries in the hope that one of these queries will elicit the desire answer, necessarily the right model to use today? Should resolver clients be more willing to terminate their search without a clear resolution of the query? Should authoritative servers be more willing to enforcing rate limitation measures and rebuff high volumes of UDP querie trains? What does "service resilience" really mean in these circumstances?

This is very much a "work in progress" story, and I am looking forward with interest to the next OARC Workshop to see how our thoughts about the DNS develops.

All the presentations from the OARC 42 Workshop can be found here: https://indico.dns-oarc.net/event/48/timetable/#20240208.detailed. Thanks to the Program Committee for a great program of presentations and the OARC staff and volunteers for making this workshop possible.

Disclaimer

Author

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

www.potaroo.net