Geoff Huston
May 2017

DNS OARC 26

OARC 26 was held in May in Madrid. Here are my impressions of the meeting, drawn from some presentations that I found to be of personal interest. All the presentations can be found at OARC's web site (https://indico.dns-oarc.net/event/26/).

Parent Delegation TTLs

The DNS seems like such a simple tool. The structure of the DNS is remarkably straightforward hierarchy name space, and the operation of resolving a name to its associated resource data is equally straightforward. But there are a considerable number of subtle aspects of this setup. One of these cornes from delegation records. When a zone is delegated, the 'parent' zone contains the names of the name servers for this delegated zone, and like all entries in a zone, it has a Time To Live (TTL) value. The child zone also contains the name servers for the zone, and associated TTL values of these records.

This presentation asked the question: Which of these two TTL values 'wins' when they differ? Parent or Child? Or does the resolver make up its own value? The basic guideline of the DNS (I hesitate to use the term "rule" as the DNS seems to be a place where rules are more like common conventions rather than absolutes!) is that the authoritative Name Servers and their TTL comes from the child zone rather than the parent, and resolvers should not make up their own value!

However, as with many aspects of the Internet, speed matters, and spending an additional query interval to get the TTL of the child NS records is often seen as an unnecessary delay and resolvers may just use the parent's NS and TTL records, and regard that as its working delegation records. Of course, this information is conventionally provided as part of the additional information section in a DNS response, and if the server is using answer minimisation then the parent delegation data will be used, as answer minimisation would strip out the child TTL. Maybe if we all adopted Child-to-Parent synchronisation in the DNS (RFC 7477) this issue of different NS records in the parent and child might go away, or at least be less prevalent!

URL: https://indico.dns-oarc.net/event/26/session/1/contribution/13/material/slides/0.pdf

DNS over TCP

The DNS uses UDP for most queries and responses. UDP's simple stateless behaviour seems to be a good match for the DNS query protocol. However, TCP is also a supported transport protocol for the DNS. TCP was designed to be used as a fallback for large responses. Prior to the use of EDNS(0) UDP Buffer Size options, the DNS protocol messages were limited to 512 byes in size when using UDP. If the DNS response is larger than 512 bytes, the responder is supposed to truncate the UDP message to stay within the 512 byte limit and the truncation flag in the response is intended to act as a signal to the querier to rephrase the query using TCP if it so wishes.

This behaviour was specified in RFC 1123, but has been recently refined with the publication of RFC 7766 in March 2016. A resolver may elect to use TCP or UDP to pass a DNS query. TCP has a higher overhead in terms of connection setup, as the parties need to complete the initial TCP handshake before the channel can be used, so one option for a resolver is to try and keep the channel open and pipeline a number of queries. The behaviour is explicitly allowed in RFC 7766, specifying that the server should only close the TCP connection after two minutes of idle time under normal circumstances.

This presentation by Jan Včelák looked at the profile of TCP connection reuse, using the DITL data from the root servers from April 2016. they found some 67 million DNS queries in this data set, in some 85 million TCP sessions. Surprisingly, some 37% of the TCP sessions asked no queries! A further 57% of TCP connections were used to make a single query. Only 6% of TCP sessions made multiple DNS queries, with most making between 2 and 20 queries, and the pattern of multiple queries appears to have some form of exponential decay. Google's DNS service makes use of multiple connections over TCP. What they also observed in the data was clients repeating the same query in the TCP session, and even cases where the client has reflected the response back to the server.

The data raises some interesting questions. Why is the number of TCP sessions with no queries so large? Can TCP improve on the reliability of the service and reduce the UDP re-query rate?

There is a serious side to this study relating to the ongoing work on DNS privacy. It seems that one part of this work is concentrating on DNS over TLS over TCP, and another emerging stream us DNS over QUIC, which is itself a form of TCP-like reliable encrypted communications with an outer wrapping of UDP, but with similar behaviours to TCP.

URL: https://indico.dns-oarc.net/event/26/session/2/contribution/12/material/slides/0.pdf

HyperLogLog applied to the DNS

One of the more captivating presentations at this meeting was by Bert Hubert of PowerDNS, applying HyperLogLog to estimating the "size" of a domain. Some registries expose these metrics, others are the subject of various forms of estimation. There are some popular sources for this data, including Keith Monshouwer's tracking of .NL, Fredric Cambus' StatDNS report, and Rick Lamb's DNSSEC reports. Obviously, the best way to count the number of delegations in a zone is to get a copy of the zone and count them. However, in the DNS it's not always as simple as this: some zones are simply not published, and this means that one has to use indirect methods to estimate the size of a zone. And here's where the HyperLogLog concept comes into play.

As Burt explained: "Imagine bowl full of marbles, with random numbers between 1 and 1 billion. A typical marble I pick may have any number, but the changes of picking a particular number should be pretty remote. However, one picks sufficient marbles, the lowest number seen is related to the number of distinct numbers in the entire bowl."

How does this relate to the DNS? Well if one takes a hash of each of the names in a zone, then this is precisely a random number set that can be processed in the same manner. So, the approach is simply to ask random questions of an NSEC3-signed zone. Each returned NXDOMAIN record provides two NSEC3 hashes of names that are in the zone.

Burt has written a tool that can measure the size of a typical NEC3-signed zone in around one third of a second. Comparing this estimate to a zone of known size shows the robustness of this technique: the .NL has 2,642,218 unique names, while the HyperLogLog estimate is 2,644,800 after 65K queries. That's within 0.2%, which is a very impressive result.

He has extended this technique to look at NSEC-signed zones, using the same technique. Even though the names are not randomised, the procedure can be coerced to produce the same estimate of domain size – which perhaps even more impressive!

A really excellent presentation, complete with pointers to the resulting zone file size report and the code.

URLs: https://indico.dns-oarc.net/event/26/session/2/contribution/6/material/slides/0.pdf
https://www.monshouwer.eu/dnssec-nl-graph/
https://www.statdns.com/
http://rick.eng.br/dnssecstat/
https://powerdns.org/dnssec-stats/
https://github.com/ahupowerdns/pdns/tree/measurensec2

Testing DNS Resolvers

One more episode in the ongoing story that the DNS is nowhere as simple as one might think was provided by Petr Špaček of nic.cz. Forwarders maintain a cache of responses, and those caches contain a large amount of state. The ordering and timing of the way the cache was populated matters, as does the previous presentation of whether the cache is using the parent TTL or the zone TTL to define cache lifetimes. There is also the issue of forwarding of resolvers, that creates as a side effect a chain of caches. The cache also needs to maintain the flags associated with the cache loading query in order to be aligned to serving the cache entry to subsequent queries. For example, an entry that fails DNSSEC validation is still viable to use if the client has set the DNSSEC Checking Disabled (CD) flag.

Application behaviour also has an impact here. The example of opening youtube.com was cited, which resulting in 15 DNS queries from a stub resolver serving the client application within 2.3 seconds. Some of these queries were sent in parallel, while it is unclear in other cases whether the application is deliberately sequencing the queries or the observed sequencing was not strictly necessary.

The work entails capturing these queries in a web test framework, with timing and query flags, and then repeating the queries varying factors such as query name minimisation, and using different stub resolvers, and using a chain of forwarders, for example. They are using a tool called "Dekard" to replay captured DNS queries and responses.

URLs: https://indico.dns-oarc.net/event/26/session/3/contribution/25/material/slides/0.pdf
https://gitlab.labs.nic.cz/knot/deckard/blob/master/README.rst

NSEC5

There is a proposed use of public/private cryptography to improve upon the current model of obfuscated denial of existence in DNSSEC. The initial problem was that a number of registries, notably Verisign's operation of the .com and .net registries, did not want to permit third parties to enumerate the contents of these two zones (exactly why they felt that zone enumeration was detrimental to their commercial interest as a registry is probably the subject of an entire article in its own right – but let's not go there right now!). The result was NSEC3 (RFC 5155) where the range field of the DNSSEC denial of existence (NSEC) record was changed to be a range of hashed values. If the hashed value of the requested label sat between the two hash values in an NSEC3 record than the label was provably absent from the zone. But because hashes are effectively a one way function (or supposed to be) then the end points in the range cannot be reconstructed from their hashed values that have been given in the response. However, hash functions are not in fact always one way functions, and with some effort they can be reversed and there are now kits to do so. Such tools enumerate NSEC3-signed zones.

NSEC5 is a new algorithm that replaces the simple symmetric hash function with a public/private key par hashing function.

The entries in the zone are sorted into a hash order using the private key to generate the hash values. Responses to queries for non-existent names require the server to generate a "proof" based on the query name and the private hash key. The returned response includes both the range indicator and this "proof. The client can validate the response by using the public part of the hash key to confirm that the query name and the proof are related. It's certainly a novel approach, but it's unclear to what extent the larger computation load in using asymmetric cryptography is viable for large scale zone servers of signed zones.

NSEC5 is implemented in the Knot and Unbound server code, supporting ECDSA with the P-256 curve. This keeps the response sizes of NSEC5 records far smaller than NSEC-3 using RSA-2048 ZSKs, which is good, but the downside is that the range values in NSEC5 cannot be pre-computed as with NSEC3, so the cost of trying to prevent third parties enumerating a zone is in compute load imposed upon the server. Seems to me that when a server must perform lot a lot of work to answer a query for a non-existent name then you are exposing the servers to DOS vulnerabilities.

URLS: https://indico.dns-oarc.net/event/26/session/6/contribution/23/material/slides/0.pdf
https://github.com/anonion0/nsec3map

Post-Quantum Cryptography

Today's digital cryptography is based on the difficultly of of two problems: one is the difficulty of finding the prime factors of a very large composite number that is the product of two primes, a problem that is a few millennia old! The other is that of discrete logarithms, which is considered to be computationally intractable. Assuming of course that the computers you are using are deterministic machines that we are used to.

But there is another area of emerging computers based not on the binary 0 or 1, but using quantum Qubits, that have values between 0 and 1. If these are connection by quantum gates to form quantum circuits that represent a quantum algorithm then some interesting properties emerge. Quantum computers that are sufficiently large can be useful in performing a search across a number space in a way that conventional computers simply cannot perform. A paper by Peter Shor changed in 1997 described "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer" which is an efficient algorithm to solve the problem posed by these forms of cryptography. It points to an approach using large scale quantum computers with quantum Fourier transform circuitry that can generate the private keys in few steps than are possible today. It seems that we are still some time away from this, but time is indeed relative and a large amount of research activity is underway to build large scale quantum computers. It's reasonable to predict that within 5 -10 years a determined and well resourced adversary would crack conventional RSA and ECDSA cryptography.

The cryptography research world is already investigating post-quantum approaches, and a number of approaches appear to have properties that do not lend themselves to an efficient solution using Shor's algorithm, but whether they withstand the scrutiny of the crypto world and are indeed relatively easy to construct and challenging to break remains to be seen.

Why are we talking about this in a DNS workshop?

Because DNSSEC.

The trust model of the collection of Certificate Authorities continues to experience failures and there are few embracing trust frameworks for the Internet outside of DNSSEC.

URL: https://indico.dns-oarc.net/event/26/session/6/contribution/4/material/slides/0.pdf

dnsprivacy.net

It's one thing to confidently expect something and quite another to have it shoved into your face! The material exposed by Edward Snowden in June 2013 was not a revelation in general terms. Communications have been intercepted for millennia and the NSA activity was just following a well-established tradition. What was a surprise was that so much of the Internet was in the clear, and the task collecting this data was in many cases quite trivial in terms of technical challenge.

One reaction from the IETF was to review the way in which encryption is folded into the application environment, and the DNS is a prime candidate for such a study. In March of 2014 the IETF formed the DPRIVE Working Group, and in August 2015 RFC 7626, DNS Privacy Considerations, was published. It may appear to be anomalous to think about security for queries relating to public DNS data, but the underlying issue here is that the DNS data may be publicly accessible, but the DNS resolution transactions that I may make should not be so readily or reliably attributed to me. Almost every Internet transaction is preceded with one of more DNS queries, and knowledge of the queries alone can provide a reasonable insight into a user's online activities.

A simplified model is that the DNS is populated by "stub" resolvers, who pass queries to "recursive" resolvers who then pass the query to relevant authoritative servers in order to respond to the query. Stub resolvers sit in the end host, so visibility of these queries can lead to knowledge of who is asking for what name. The recursive resolver is slightly more opaque, in that the end client information is not usually attached to the query, unless of course the recursive resolver has enabled EDNS(0) Client Subnet. The stub resolver cannot force the recursive resolver to turn off this option.

The major effort of the DPRIVE Working Group has been the stub to recursive resolver step. In May 2016 RFC 7858, the specification of DNS over TLS, was published. The Unbound resolver has been used to support a test DNS over TLS recursive resolver, and Bind has been used in conjunction with Oddly enough, DNS over TLS specified the use of TCP port 853, while a more logical approach in terms of enhanced privacy would be to reuse port 443, as part of an effort to further obfuscate the DNS traffic and 'hide' it in the data stream of HTTPS secure web transactions.

Attention is now shifting to the queries made to authoritative servers. One of the first major steps here is query name minimisation, only passing as many labels in a query to the authoritative name server as is required to establish the name servers of the next level delegated zone.

As good as all this sounds, it is not the entire story. It may be that DNS query data is gathered by covert means of wiretapping, unauthorized system intrusion, or data theft, but these days data is just another commercial commodity so it should not be surprising to learn of DNS recursive resolvers and name servers of all kinds that simply sell their DNS query stream to interested buyers. Why go to all the trouble to covertly steal data when it is openly for sale in the first place? And why are we attaching such importance to encryption on the wire when the other end of the query may be simply selling the very same data to anyone willing to pay?

URL: https://indico.dns-oarc.net/event/26/session/4/contribution/10/material/slides/0.pdf

DNS over Foo

The DNS over TLS work has prompted a more general investigation of DNS over transport systems other than UDP.

It would be good if we could use a UDP-styled encryption transport, and use something like "Datagram Transport Layer Security" (RFC 6347). There is a specification for DNS over DTLS, but there are a couple of wrinkles with DNSSEC and responses. To quote from the specification "The DNS server MUST ensure that the DNS response size does not exceed the Path MTU, i.e., each DTLS record MUST fit within a single datagram, as required by [RFC6347]." Responses that require some level of packet fragmentation is to be expected from time to time when DNSSEC is used. For example, a query for the DNSKEY value of the .org domain generates a response with four keys and three signatures, with a total response size of 1,625 octets. It appears that DNS over DTLS has an explicit maximum response size of less than 1,500 octets, and larger DNS responses require truncation and fallback to, presumably, DNS over TLS. At this stage, there are no known implementations of DNS over DTLS, despite the specification being published as an RFC in February of this year (RFC 8094). Both DNS over TLS and DNS over DTLS are useful in the context of a stub resolver communicating with a recursive resolver as a means of preventing on-the-wire eavesdropping of DNS queries.

Another approach is DNS over HTTPS, or more specifically DNS over HTTP/2 over TLS over TCP. This is again suitable for a stub resolver communicating with a recursive resolver, but is more suited to embedded stub resolvers within browsers, or javascript apps running within a browser environment.

More recent is the proposal for DNS over QUIC (draft-huitema-quic-dnsoquic). This approach provides the same DNS privacy protection as DNS over TLS, including the option for the client to authenticate the server by means of an authentication domain name. Like DNS over TCP or DNS over TLS, DNS over QUIC provides an improved level of source address validation for DNS servers compared to DNS over UDP. The QUIC transport is a stream-based transport, so it is not constrained by path MTU limitations on the size of DNS responses it can send. The DNS traffic follows a simple pattern in which the client sends one query over a selected QUIC stream, and the server provides a response over the same stream. Multiple queries use multiple QUIC streams. There are no known open implementations of DNS over QUIC as yet, but as with QUIC itself, there is a lot of interest in this topic.

It is also possible to take a more conservative stance with QUIC and use DNS over HTTP/2 over QUIC, in which case there are no changes to the QUIC profile itself. Why QUIC? It runs over UDP as a user space implemented transport protocol. It uses transport level encryption to encrypt not only the payload, but also the QUIC stream parameters and all other flow controls. QUIC multiplexes multiple streams on a single connection, and allows for version negotiation to allow for protocol evolution. QUIC has no Head-of-line blocking, up to date congestion control protocol, it is resilient to NAT-rebinding, and it supports rapid 0-RTT session resumption (like TLS 1.3), which is ideal in a stub to DNS recursive resolver scenario.

URLs: https://indico.dns-oarc.net/event/26/session/8/contribution/32/material/slides/0.pdf
https://indico.dns-oarc.net/event/26/session/8/contribution/33/material/slides/0.pdf

ANAME

It appears that these days the concept of the DNS providing a simple name-to-address mapping is just not enough, and what we are after is a generic name and various forms of on-demand resolution that map the name through various forms of indirection. There are various reasons why one would want to have the DNS resolution process translate one DNS name to another, and various ways to do it.

One is the CNAME record (in this case 'C' is for canonical name), which replaces one fully qualified DNS name for another. There are some exclusions in the use of CNAMES, including that MX and NS records must not point to a CNAME alias, and if a name is defined in a CNAME then no other DNS resource records are permitted to be associated with that name (other than a DNAME records and the

DNSSEC RRSIG and NSEC(3) records). It's often used in the content hosting area, where the client's name is translated by a CNAME into a hosting record.

Another is the DNAME record. Whereas the CNAME record applies to a single name that is an exact match between the queried name and the name to which the CNAME is applied, the DNAME record creates an alias for an entire subtree, so that any subdomain of the name specified as a DNAME'd name is translated into a subdomain of the nominated DNAME. A DNS server that serves a DNAME will also synthesise a CNAME in its response. A DNAME is equivalent to a CNAME applied individually to all names in an entire subtree of the DNS.

In the broader DNS usage, name redirection and aliases can be performed on the resolver side through NAPTR and SNAPTR records as well as SRV records.

Do we have enough tools for dynamic server-side name translation? Well, evidently not!

Into this mix comes the ANAME record proposal (draft-hunt-dnsop-aname). It's a server-side indirection pointer with an exact match requirement, much like a CNAME, but it is only invoked when the query is for an IP address (A or AAAA record). That means that an ANAME record can co-exist with other resource records, so that it can be used at the apex of a zone as well as within the zone. Authoritative servers configured with ANAME records will answer address queries for the ANAME owner with addresses found at the ANAME's target, and also provide in the response the ANAME itself. What this construct allows is redirection only in the address record query, but leaving all other attributes of a name unaltered.

URL: https://indico.dns-oarc.net/event/26/session/8/contribution/34/material/slides/0.pdf

The DNS Architecture Revisited

A presentation on the topic of "The Importance of Being an Earnest Stub" raised some generic questions about the architecture of the DNS. The conventional model of the DNS is that of a stub resolver passing queries to a caching recursive resolver that, in turn, queried authoritative servers. All the work is performed by the recursive resolver in this scenario. With the introduction of DNSSEC into the mix then the recursive resolver signals back to the stub that it has successfully validated the answer by setting a bit in the response. There are number of problems with this model, not the least is hijacking of the recursive resolver so that a bad response is passed back to the stub resolver. How can we improve on this model of credulous trust in the unprotected path between the stub resolver and its selected recursive resolver?

One possible measure is for the stub resolver to perform its own DNSSEC validation. This too can be through queries made to via the recursive resolver, but now the stub is not relying on the validation being performed by the recursive resolver, but is validating these responses for itself.

Another measure is the protection of the path between the stub and the recursive resolver. While this protects to some extent against man-in-the-middle interception and eavesdropping attacks, it is not an absolute level of protection, and it does imply that the stub places complete trust in the integrity of the recursive resolver - which is perhaps again a measure that may, at times, be a stretch of reasonable credibility.

If we are using TLS then this relies on CA integrity to protect the certificate offered in the TLS startup exchange. CA's have an unfortunately marred reputation for integrity, and perhaps one should also add DANE as a measure to validate the key being offered through TLS. But if this is being used as a bootstrap of a secure TLS session with the recursive resolver then how does a stub perform treated interactions with the DNS in order to perform the DANE TLSA lookups? One promising approach is

described in draft-oetf-tls-dnssec-cahin-extension, where the validation chain for the TLSA record is provided as an extension to the TLS certificate.

While it is tempting to eschew the use of a recursive resolver completely and operate the stub resolver as a fully functional DNS resolver that directly queries authoritative name servers, there are some drawbacks to such an approach. The first is that the stub resolver cannot effectively utilise the queries made by others to take advantage of a local cache. Such a standalone resolver will probably take longer to resolve names. The stub is also unable to be able to make s secure encrypted session with all (or even any at this stage) authoritative servers, which implies that all of its queries are in the clear.

It seems that a prudent stub resolver should make a secure association with a recursive resolver that it is prepared to trust if it wants some protection from eavesdropping of DNS queries. To what extent it is prepared to take all answers on trust from the recursive resolver and to what extent it wishes to separately validate the signed answers it receives is up to the stub. The additional cost of such additional prudence is of course additional resolution time, which some applications may be an acceptable cost, while others may view this as being overly cautious.

URL: https://indico.dns-oarc.net/event/26/session/4/contribution/24/material/slides/4.pdf

Conclusion

DNS OARC meetings are a very concentrated dose of DNS in two packed days. For many attendees however, too much DNS is barely enough and there is much we can still do to improve the resilience, robustness, efficiency and versatility of the DNS.

It may be that according to classical computer science any problem can be solved by adding another layer of indirection, but where Internet engineering is involved, any problem can be solved more effectively by just stuffing it into the DNS!

Author

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

www.potaroo.net

Disclaimer

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