The ISP Column
A monthly column on things Internet

Geoff Huston
November 2016

Scoring the DNS Root Server System

The process of rolling the DNS Root's Key Signing Key of the DNS has now started. During this process there will be a period where the root zone servers' response to a DNS query for the DNSKEY resource record of the root zone will grow from the current value of 864 octets to 1,425 octets. Does this present a problem?

Let's look at the DNS Root Server system and score it on how well it can cope with large responses. It seems that awarding stars is the current Internet way, so let's see how many stars we'll give to the Root Server System for their handling of large responses.

Packets and Networks

What is it about large responses that are an issue here?

There are a number of persistent themes in packet networking that appear to be unresolved despite many decades of experience. One of these is the handling of packet sizes.

Packet-switched networks dispensed with the constant time base used in time-switched networks. Instead, they allow individual packets to be sized according to the needs of the application as well as the needs of the network. Smaller packets have a higher packet header to payload ratio, and are consequently less efficient in data carriage. On the other hand, within a packet switching system the smaller packet can be dispatched faster, reducing the level of head-of-line blocking in the internal queues within a packet switch and potentially reducing network-imposed jitter as a result. Larger packets allow larger data payloads which in turns allows greater carriage efficiency. Larger payload per packet also allows a higher internal switch capacity when measured in terms of data throughput. But larger packets take longer to be dispatched and this can be a cause of increased jitter.

Some packet network designs, notably ATM, used a constant-sized packet, replicating many of the properties of the time-switched systems. Others deferred the decision over packet size to the next layer up in the protocol stack and supported variable packet sizes. Ethernet, designed in mid-1970's, adopted a variable packet size, with supported packet sizes of between 64 and 1,500 octets. FDDI, a fibre ring local network used a variable packet size of up to 4,478 octets. Frame Relay used a variable packet size of between 46 and 4,470 octets. The choice of a variable-sized packets allows to applications to refine their behaviour. Jitter and delay-sensitive applications, such as digitised voice, may prefer to use a stream of smaller packets to attempt to minimise jitter, while reliable bulk data transfer may choose a larger packet size to increase the carriage efficiency. The nature of the medium may also have a bearing on this choice. If there is a high bit error rate (BER) probability, then reducing the packet size minimises the impact of sporadic errors within the data stream, which may increase throughput.

The real issues surface when you impose an overlay end-to-end network transport design on top of these various packet delivery media. What should an Internet router do with a 4,478 octet IP packet received on an FDDI interface when the next hop is an Ethernet segment with a maximum packet size of 1,500 octets? The answer varies according to the IP version. In IPv4, as long as the DON'T FRAGMENT bit in the IP packet header is clear, the router is permitted to split the payload across several IP packets, fragmenting the packet to match the next hop maximum message size, replicating the IP header in all the fragments (aside from the fragmentation control header fields, of course). The IPv6 behaviour is similar to that of IPv4 when the packet header has the DON'T FRAGMENT field set to 1. The router is not permitted to fragment the packet, and as it cannot forward the packet onward, an ICMP diagnostic message (containing the leading octets of the to-be-discarded packet) is sent backwards to the source address in the original packet and the packet is then dropped.

In the IPv4 router-fragmentation case nothing more need be done. Fragmentation is handled at the IP layer and the reassembled complete packet is delivered to the upper layer transport protocol at the other end. But in the other case, namely IPv6 and IPv4 when the IPv4 DON'T FRAGMENT field is set on, then the issue of a path packet size issue needs to be handled at the transport layer. For TCP the intended response by the sender is to be passed the ICMP diagnostic packet and have the session reduce its MSS value to match the reduced size. TCP will then assume control for repairing the data gap because of the dropped packet and the session should continue. For UDP it's a little trickier. UDP has no "memory" so the received ICMP diagnostic message has no logical delivery point within the local host. Ultimately, this becomes a problem at the application layer, and the application using UDP has to detect packet loss and to take into account a potential cause of packet size mismatch in its recovery behaviour. If the application is lucky the host will lend a hand here and place a host entry in its local forwarding table that records the original destination address and the maximum packet size that can be sent too that address, based on the size field contained in the packet too big ICMP diagnostic message.

Why is all this relevant?

Because DNS.

DNS

The DNS is a UDP application, and in the context of the Internet its a critical application. Pretty much every transaction across the Internet starts with a name-based rendezvous, and the first step is to resolve the name to an IP address. For this we use the DNS.

The original design of the DNS limited packet payloads using UDP to 512 octets (RFC1035). Interestingly, the motivation behind this appeared not to be the desire to avoid packet fragmentation per se, but to avoid a different packet size issue: the maximum reassembled packet size that an IPv4 host is assured to be able to reassemble is 576 octets (RFC791). This limitation has some interesting side-effects. For example, this size limitation means that the number of authoritative name servers listed in response to a DNS priming query was limited to 13 entries, as long as you only wanted to know the IPv4 addresses of these servers. A 14th entry would push the DNS response to list the root zone's name servers over 512 octets in length. From this came the limitation of 13 distinct root name servers for the DNS.

Some things have changed over the intervening years, but interestingly the 512 octet limit to DNS payloads, in theory at any rate. This is despite the observation that a new informal standard packet MTU size has been adopted by the Internet: these days a 1,500 octet packet stands a strong chance of being passed through the Internet unscathed. But while 1,500 octet packets stand a good chance of making it through, there is a difference between a probabilistic estimate and certainty. IPv4 actually provides no certainty in this space. While the maximum size of a reassembled IPv4 packet was specified at 576 octets, the minimum unfragmented size was not. The IPv6 specification defines a minimum unfragmented IP packet length of 1,280 octets. That is, any IPv6 packet with a size equal to or less than

1,280 octets will not be fragmented by an IPv6 carriage network, and will be accepted by the intended host.

Why 1,280 octets? It seems like such an arbitrary number. The answer I've been given is that 1,280 is what you get when you add 1,024 and 256! This somewhat meaningless piece of maths was intended to ensure that an IPv6 packet could transit an underlying Internet fabric that presumably supported 1,500 octet packets, and also admit the possibility of a number of levels of IP-in-foo encapsulation. Personally I find this arbitrary choice to one of the major design flaws of IPv6 and the cause of much brokenness in the IPv6 Internet!

While the 512 octet limit still applies to the DNS in theory, it's not uncommon to see larger DNS responses being pushed around the Internet with apparent impunity. This is particularly the case when DNSSEC is added, and the response contains the digital signature as well as the requested data. To cope with this, the DNS protocol now uses an extension mechanism, EDNS(0), defined in RFC6891, that allows a querier to specify the largest UDP response it is willing to receive. If this number exceeds 1,500 octets (and it is commonly set to 4,096 in many resolvers), then it is highly likely that the DNS UDP response will be fragmented, and the querier will need to reassemble the IP fragments in order to assemble the DNS response. If the response would be larger than the offered EDNS(0) buffer size, then the response will necessarily be truncated to fit within the specified payload size. If the querier wants the complete answer it will either need to re-query with a larger EDNS(0) buffer size, or, more commonly, re-query using TCP.

Again, why is this relevant?

Because DNS, DNSSEC and the forthcoming roll of the KSK of the root zone.

DNS Large Responses

DNS resolvers that perform DNSSEC validation will, from time to time, query toward one of the root zone name servers for the signed valued of the root zone's DNSKEY records. When there is no key roll happening, the response contains onee KSK, one ZSK and one RRSIG signature signed by the KSK. Now that the ZSK is 2,048 bits in size the total size of this DNS response is 864 octets in length.

During the planned roll of the KSK of the DNS Root zone there will be a period when two KSKs (old and new) are in the root zone at the same time, and the DNSKEY record will be signed by both of these KSK keys. The signed response to a query for the root zone's DNSKEY record will inflate from 864 octets to 1,425 octets at this point in time. In the current plan this will occur on the 11th January 2018, and last for 20 days (http://www.slideshare.net/apnic/rolling-the-root-zone-dnssec-key-signing-key, slide 28)

As far as I am aware, this is the first time a 'normal' DNS response from the root servers will exceed 1,232 octets in length, and the IPv6 UDP packet will exceed 1,280 octets in length.

Now in theory this should not present a problem, but theory and practice often tend to diverge.

DNS Root Servers and Large DNS Responses

How will the root servers deliver this response?

These days with anycast constellations any question about root server behaviour is not a simple question. Let's simplify this a bit and ask what can we see from the root servers from one particular vantage point?

By crafting a relatively long query name for a non-existent domain name we can get a root server to generate a response where the DNS payload is 1,268 octets in length. In this case the query used EDNS(0) and specified a UDP buffer size of 4096, and requested DNSSEC signatures to be attached to the response. An IPv6 UDP datagram containing that response is 1,316 octets long and the IPv4 UDP datagram is 1,296 octets long. What we see from each root server is shown in Table 1.

Root	**IPv4**			**IPv6**		
	Truncate	**Fragment**	**TCP MSS**	**Truncate**	**Fragment**	**TCP MSS**
A	N	N	1,460	1,280	N	1,440
B	1,280	N	1,460	1,280	N	1,440
C	N	N	1,460	N	N	1,440
D	N	N	1,460	N	N	1,440
E	N	N	1,460	N	N	1,440
F	N	N	1,460	N	1,280	1,440
G	1,280	N	1,460	1,280	N	1,440
H	N	N	1,460	N	N	1,440
I	N	N	1,460	N	N	1,440
J	N	N	1,460	1,280	N	1,440
K	N	N	1,460	N	N	1,440
L	N	N	1,460	N	N	1,440
M	N	N	1,460	N	1,280	1,440

Table 1 – Root Server Response Profile to a large DNS response

Table 1 shows that in IPv4 11 of the 13 root servers send the 1,296 octet UDP response packet directly to the querier. The other two root servers, B and G, elect to respond with a shortened (truncated) response. Some experimentation with varying response lengths shows that this truncation occurs when the DNS response is 1,252 octets or larger. The querier is implicitly directed to retry using TCP through this response. This would tend to suggest that the root server is attempting to limit its responses to be no more than 1,280 octets in length, even in the case of IPv4 responses. However, on both B and G, a TCP session offers an MSS of 1,460, indicating that both root servers appear to have a local MTU setting of 1,500 octets. For IPv4 this is entirely unexpected behaviour, and it is unclear why B and G have chosen to configure their IPv4 behaviour to perform response response truncation in this manner, as it seems to be inviting extraneous TCP sessions to be set up.

For IPv6 7 of the 13 root servers send the 1,316 octet UDP response packet without fragmentation. F and M elect to fragment the response, fragmenting the packet so as to fit within a 1,280 octet limit. A, B, G and J elect to truncate the response instead. When opening a TCP session all four root servers offer an MSS of 1,440 octets, indicating that they are using a 1,500 octet MTU for TCP over IPv6. This behaviour seems to be more than a little odd.

The aim of the root servers is to maximise the likelihood that the response will be received by the recursive resolver, and in so doing it needs to chart a careful course between the various operational pitfalls that we are aware of.

Let's look at UDP first.

In IPv4 there is a problem with firewalls allowing fragments through, and some recursive resolvers live behind firewalls that discard trailing fragments of a fragmented packet. For this reason, it makes a lot of sense to use a 1,500 octet value for the maximum IP packet size for UDP over IPv4, so as to avoid gratuitously fragmenting outbound IPv4 UDP packets at the source.

A similar line of reasoning holds in IPv6, but the problems with fragmentation are now twofold. Not only are firewalls prone to discard trailing IPv6 fragments, but certain routers are prone to discard all fragmented IPv6 packets. This is due to the use of an extension header in IPv6 to carry the IP fragmentation control fields. Some commonly deployed routers discard all IPv6 packets that contain IPv6 extension headers, including fragmentation extension headers. This has been observed to affect recursive resolvers. Some 30% of users that sit behind IPv6-capable resolvers use resolvers that are seen to be unable to receive a fragmented IPv6 packet. The F and M root servers fragment the IPv6 packet as if the server used a 1,280 octet MTU. This is not optimal behaviour in the light of this widespread level of packet mis-handling.

The other option instead of fragmentation is perform response truncation.

In IPv4 the B and G servers do not deliver a large UDP response, even when the query specifies a large UDP buffer size. Instead, the server truncates the response so as not to deliver a UDP datagram larger than 1,280 octets.

In IPv6 A and J join B and G in truncating the IPv6 response as if there was a local MTU size of 1,280 octets. The intention here is to push the client resolver into re-issuing the query over TCP. So, how does TCP work with the root servers?

Firstly, TCP is not a viable option for all resolvers. In fact previous measurements (http://www.potaroo.net/ispcol/2013-09/dnstcp.pdf) have shown that 17% of resolvers that query authoritative name servers appear to be unable to perform a query using TCP. This inability of the resolver to perform a TCP query is either due to some local resolver configuration, or an overly zealous firewall front end that assumes that the DNS is exclusively a UDP-based protocol. The result is that just under 3% of clients are affected by this and cannot resolve a name where the UDP response is truncated. So TCP has its problems for the DNS.

What happens when the resolver is capable of performing a TCP DNS query?

In IPv4 all the root servers offer a TCP MSS of 1,460 octets. This is indicative of a local MTU setting of 1,500 octets in IPv4 for TCP. This appears to be a robust choice.

In IPv6 all the root servers offer a TCP MSS of 1,440 octets. Again, this is indicative of a local MTU setting of 1,500 octets. However, in this case I would offer the view that this is a sub-optimal local configuration. The problem lies when the path includes some form of Path MTU Black Hole. In IPv6 sending a TCP packet that is too large for the path results in an ICMP6 message being sent back to the host. If the host receives the diagnostic message, then it is in a position to drop its session maximum segment size and resend the packet. If the ICMP6 PTB message is lost or filtered before it reaches the original packet's sender then the TCP session is wedged and cannot proceed. The conservative workaround for this is to avoid the packet too big situation altogether. If the sender were to set the TCP MSS for IPv6 down to 1,220 octets, then no TCP packet would be larger than 1,280 octets and the packet would not require fragmentation (at least if everyone honours the 1,280 MTU limit in the IPv6 transit networks). At the response sizes we are talking about for the root servers the marginal speed increases seen in raising the MSS from 1,220 to 1,440 is negligible, while the consequent Path MTU blackholing is significant.

Scoring the Root Servers

How can we score the actions of each root server when dealing with a response that's larger than 1,280 octets?

- If the IPv4 UDP packet is sent without fragmentation for packets up to 1,500 octets in size, then let's give the server a star.
- If the offered IPv4 TCP MSS value is 1,460 octets, then let's give the server another star.
- If the IPv6 UDP packet is sent without fragmentation for packets up to 1,500 octets in size, then let's give the server a star.
- If the IPv6 UDP packet is sent without truncation for IPv6 packet sizes up to 1,500 octets, then let's give the server a star.
- If the offered IPv6 TCP MSS value is no larger than 1,220 octets, then let's give the server another star.

How do the root servers fare on this five star rating system? Again, I should repeat that this is the results from a test performed from just one vantage point in the Internet. It could be that different anycast instances of these root servers have different behaviour, however such variation in behaviour in an anycast situation would make some tasks, particularly related to diagnosing failure, far worse, so let's assume that the sane thing is going on here and all anycast instances are essentially the same in this respect.

A	IPv4 is good. IPv6 truncates UDP at 1,280 octets and offers an IPv6 TCP MSS of 1,440.	Could do better	✋✋✋
B	IPv4 truncates UDP at 1,280. IPv6 truncates UDP at 1,280 octets, and offers an IPv6 TCP MSS of 1,440.	Epic fail!	
C	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
D	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
E	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
F	IPv4 is good. IPv6 fragments UDP at 1,280 octets, yet it offers an IPv6 TCP MSS of 1,440.	Pretty ordinary	✋✋
G	IPv4 truncates UDP at 1,280. IPv6 truncates UDP at 1,280 octets, and offers an IPv6 TCP MSS of 1,440.	Epic fail!	
H	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
I	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
J	IPv4 is good. IPv6 truncates UDP at 1,280 octets, and offers an IPv6 TCP MSS of 1,440.	Could do better	✋✋✋
K	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
L	IPv4 is good. IPv6 UDP uses a 1,500 octet MTU, but it offers an IPv6 TCP MSS of 1,440.	Almost there	✋✋✋✋
M	IPv4 is good. IPv6 fragments UDP at 1,280 octets, and offers an IPv6 TCP MSS of 1,440.	Pretty ordinary	✋✋

Table 2 – Rating the Root Servers

By this metric, the average for the entire root server system is 3 out of 5 stars, which is passable, but not exactly inspiring.

If we split out IPv4 and IPv6, the average IPv4 score is 1.7 out of 2 stars, whereas the average IPv6 score for the root server system is 1.5 out of 3 stars.

This is a somewhat disappointing outcome. We are talking about the servers for the DNS root zone, and when a DNSSEC-validating recursive resolver cannot prime its local state with the ZSK state of the root zone through DNS queries, then the resolver simply cannot function. So, in some sense, failure is not an option here, yet the settings we see in the DNS root zone's servers, particularly for IPv6, elevate the odds of encountering failure when the response being managed is one that sits in that twilight zone between 1,280 and 1,500 octets in length. And in a little over a year from now that's exactly what will be happening in the root zone of the DNS.

However, the real question is what this behaviour implies for users. How many users will be stranded from the entire DNS root name system for the period when a large response is an intrinsic part of anchoring a validating DNS resolver into the global DNS? Earlier work has suggested that this count sould be a relatively small number, but perhaps we should revisit this result in the light of this additional information about exactly how root servers behave when sending large DNS responses.

But that's best left as a question for another day and another article.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990's. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001 and chaired a number of IETF Working Groups. He has worked as an Internet researcher, as an ISP systems architect and a network operator at various times.

www.potaroo.net

Disclaimer