November 2023
Geoff Huston

IPv6, the DNS and Happy Eyeballs

There was a draft that caught my attention during DNSOPS Working Group session at the recent IETF 118 meeting on the topic of "DNS IPv6 Transport Operational Guidelines". This draft proposes to update an earlier guideline document with some new guidelines.

The original document, RFC3901, titled "DNS IPv6 Transport Guidelines", is quite short, and the basic message is:

```
In order to preserve name space continuity, the following
administrative policies are recommended:

   - every recursive name server SHOULD be either IPv4-only or dual
     stack,

      This rules out IPv6-only recursive servers.  However, one might
      design configurations where a chain of IPv6-only name server
      forward queries to a set of dual stack recursive name server
      actually performing those recursive queries.

   - every DNS zone SHOULD be served by at least one IPv4-reachable
     authoritative name server.

      This rules out DNS zones served only by IPv6-only authoritative
      name servers.

Note: zone validation processes SHOULD ensure that there is at least
one IPv4 address record available for the name servers of any child
delegations within the zone.
```

The draft that proposes to update this guidance is nowhere as clear or succinct as RFC3901. Buried in the verbiage of draft-momoka-dnsop-3901bis is the guidance that:

```
it is RECOMMENDED that at least two NS for a zone are dual stack name
servers.

Every authoritative DNS zone SHOULD be served by at least one IPv6-reachable
authoritative name server
```

I must admit that I'm very uncomfortable with this level of advice in the content of using the DNS over IPv6 transport in such strongly worded terms. This recommendation obviously plays well to the set of highly vocal IPv6 zealots out there, but aside from such unthinking zealotry and unfounded opining, there is some room for doubt here. Is this strongly worded direction in this draft a sensible recommendation for the DNS at this point in time?

Has anything changed other than the elevated level of hardline theocratic opinion from the IPv6 pews since the publication of RFC3901 in 2004 to warrant the elevation of IPv6 support in the recursive-to-authoritative DNS to a "SHOULD" in this draft? Let's have a look.

SHOULD and RECOMMEND

Firstly, a word or two about terminology, because words matter. One of the more frequently cited RFCs within the entire set of RFC documents published by the IETF is RFC 2119. It's a definition of *key words* to use in RFCs, authored by Scott Bradner in 1997. The document contains a set of words that, in my view, have become the most commonly abused words in the world of RFCs.

This RFC document defines the intent of the terms MUST, MUST NOT, SHOULD, SHOULD NOT and MAY. They definitely SHOULD NOT be used casually in RFCs. As RFC2119 says:

```
Imperatives of the type defined in this memo must be used with care and
sparingly.  In particular, they MUST only be used where it is actually
required for interoperation or to limit behaviour which has potential for
causing harm (e.g., limiting retransmissions)  For example, they must not
be used to try to impose a particular method on implementors where the
method is not required for interoperability.
```

And also:

```
3. SHOULD   This word, or the adjective "RECOMMENDED", mean that there
   may exist valid reasons in particular circumstances to ignore a
   particular item, but the full implications must be understood and
   carefully weighed before choosing a different course.
```

For me, that says that one SHOULD NOT use the term SHOULD when the RECOMMENDED behaviour can be a source of harm, where harm includes needless retransmissions. Yet needless retransmissions and heightened likelihood of resolution failure is precisely the operational problem we see with the use of IPv6 as the UDP substrate for DNS responses in the recursive-to-authoritative DNS realm.

IPv6, UDP and Large DNS Responses

The issue here is that using DNS over UDP over IPv6 trips up over one of the very few protocol adjustments that were made in the IPv6 specification from IPv4, and in the case of large DNS responses over IPv6 the reliability and efficiency of passing large payloads over UDP and IPv6 falls as a consequence of this design decision in IPv6.

The underlying issue is that IPv6 does not permit *forward fragmentation*. It's as if the IPv4 DON'T FRAGMENT bit has been set ON all the time. If an IPv6 packet is too large to be forwarded onto the next link the IPv6 router must generate an ICMP6 Packet Too Big message addressed to the source address of the original packet. Upon receipt of this message, the original sender should cache this revised MTU size for that particular destination address for a period (commonly some 15 minutes). What this message does not do is trigger retransmission of the packet using packet fragmentation. That is left to the upper layer protocol to detect the packet drop and perform some form of recovery, if it can. Now for TCP this recovery will be the case, as the lost packet will trigger TCP to retransmit the lost packet, and the revised destination MTU should cause either the TCP session to recalculate the session MSS value or allow the outgoing TCP packet to be fragmented at source to match the new MTU size. However, no such transport protocol retransmission is available for UDP. The offending large packet never gets resent, and it is left to the upper-level DNS application to timeout and for the DNS client to repeat the query to the DNS server.

This has some consequences and undermine the robustness of this process:

- Firstly, the DNS timers are very conservative as the DNS does not maintain round trip estimates, so implementations use a static timer to determine if a response has been dropped. If these are too short, then the DNS client will needlessly resend queries and potentially swamp the DNS server. If they are too long, then this will impact the elapsed time to resolve a DNS name.

Different DNS implementations have chosen different values for the timeout interval, and commonly seen values are 375ms, 400ms and 1second.

- Secondly, ICMPv6 packets are commonly filtered as a security risk. They are essentially unverified packets and malicious injection of such packets could be used to disrupt a communications session. If the ICMPv6 Packet Too Big never makes it back to the source, then the timeout and repeat will suffer the same fate.

- Thirdly the use of DNS anycast can disrupt this signalling. It is feasible for the ICMPv6 message to be sent to the wrong server.

- The IPv6 problems with packet fragmentation don't stop there. The IPv6 protocol design uses the concept of an optional set of *extension headers* that are placed between the IP header and the UDP header. This means that the UDP transport header is pushed back deeper into the packet. Now the theory says that this is irrelevant to the routers in the network as the network devices should not be looking at the transport header in any case. The practice says something entirely different. Many paths through the Internet today use load distribution techniques where traffic is spread across multiple carriage paths. In order to preserve the order of packets within each UDP or TCP flow the load distributor typically looks at the transport header to keep packets from the same flow in the same path. Extension headers and packet fragmentation make this a far more challenging task to perform at speed as the transport header is no longer at a fixed offset from the start of the header, but at a variable location based on unravelling the extension header chain. Some network devices push such packets away from the ASIC-based fast path and queue them up for CPU processing. Other devices simply discard IPv6 packets with extension headers. This happens at quite significant levels in the IPv6 network (https://stats.labs.apnic.net/cgi-bin/v6frag_ccpage?c=XA)

  It appears that UDP packet fragmentation and IPv6 don't go well together. So perhaps we should just avoid fragmented DNS packets altogether in IPv6. However, that might be easy to say, but it's harder to implement. It's all too easy to find large DNS responses in UDP, particularly if DNSSEC is being used.

When the client times out due to non-reception of the response to its query, it may repeat the query. If the ICMPv6 message has been generated and received, then the outgoing response packet will be fragmented according to the revised MTU value. We then may encounter the next issue, namely that many systems consider fragmented packets to represent an unacceptable security risk and are often dropped. Fragmentation itself in the DNS is best avoided (https://datatracker.ietf.org/doc/draft-ietf-dnsop-avoid-fragmentation/)

In contrast, IPv4 handles large responses in a slightly different manner. Large UDP packets in IPv4 can be fragmented in flight. There is no need to send a reverse signal back to the packet sender and no necessary need for the client to timeout and re-query. Of course, all this assumes that fragmented packets will reach their intended destination, and will be successfully reassembled at the destination, but aside from the use of fragment filters, fragmented IPv6 packets add the additional complication of the IPv6 Fragmentation Extension Header, which have their own problems of IPv6 extension header dropping in networks.

So large payloads in the DNS and the use of UDP transport protocol create a far less efficient outcome in IPv6 as compared to IPv4.

I've looked at this topic in some depth in July 2020 (https://www.potaroo.net/ispcol/2020-07/dns6.html), so I will not repeat the detailed outcomes of that study here, but the conclusion is important:

```
In a measurement performed at the end of April 2020 we performed this
experiment some 27M times and observed that in 11M cases the client's DNS
systems did not receive a response. That's a failure rate of 41%

How well does IPv6 support large DNS responses? Not well at all, with a
failure rate of 41% of user experiments.
```

Clearly things within the DNS will need to change if we ever want to contemplate an IPv6-only DNS, as such a significant failure rate for large DNS responses over IPv6 renders IPv6 as simply not viable.

What causes large DNS responses?

DNSSEC.

There has been a significant push in DNSSEC circles to evolve from prime-number based crypto algorithms to elliptical curve algorithms as they offer shorter keys and signatures as compared to prime-number cryptography of equivalent strength. Indeed, with ECDSA P-256 and by using some subtle changes in the denial of existence responses it's possible to ensure that all DNSSEC responses will comfortably fit in a payload of under 1,400 bytes. But, of course, this is assuming that we will never need to shift to a post-quantum computing world.

So, we could ditch DNSSEC and contemplate a robust DNS operating over IPv6 using UDP. Or if we want to keep DNSSEC then the considerations of efficiency and robustness of the DNS tend to say we better have IPv4 at hand and use it as a preferred protocol, and IPv6 is simply an option. Which is what RFC3901 already states.

Current Behaviours in the DNS

On one day, the 9th November 2023, the APNIC Ad measurement system collected some 46,447,008 distinct query names/query type couplets for a single category of DNS measurement. The system collected the sequence of queries and timing of these queries that ensued in resolving this query name. The DNS query names contained uniquely generated label components, so there was no DNS caching that would assist in the resolution function and all queries were passed to the authoritative nameserver, where the queries were recorded.

In this work we will not count the actions of individual recursive resolvers but look at the total behaviour. The reason for this choice is that not all resolvers are equal, and a resolver with just a single client should not have the same weight in an aggregated measurement as a resolver service with a few million clients. By measuring DNS behaviour using query sequences, we are in effect performing a query-weighted view of resolvers, weighting the measurements of each resolver's behaviour by the intensity of use of that recursive resolver.

The basic question here is whether recursive DNS resolvers perform "Happy Eyeballs" when resolving a name using authoritative name servers. Here's the current definition of this behaviour as it relates to the DNS:

Section 3.1 of RFC 8305, Happy Eyeballs v2 reads:

```
3.1.  Handling Multiple DNS Server Addresses

   If multiple DNS server addresses are configured for the current
   network, the client may have the option of sending its DNS queries
   over IPv4 or IPv6.  In keeping with the Happy Eyeballs approach,
   queries SHOULD be sent over IPv6 first (note that this is not
   referring to the sending of AAAA or A queries, but rather the address
   of the DNS server itself and IP version used to transport DNS
   messages).  If DNS queries sent to the IPv6 address do not receive
   responses, that address may be marked as penalized and queries can be
```

```
sent to other DNS server addresses.

As native IPv6 deployments become more prevalent and IPv4 addresses
are exhausted, it is expected that IPv6 connectivity will have
preferential treatment within networks.  If a DNS server is
configured to be accessible over IPv6, IPv6 should be assumed to be
the preferred address family.
```

Between recursive resolvers and authoritative DNS servers there is a visible preference to start the DNS resolution process using IPv4. Some 65% of these 46M query sequences started with a query over IPv4, so clearly, in terms of Happy Eyeballs conformance, we are not off to a good start! It is unclear if this Happy Eyeballs advice in RFC8305 relates only to the edge of the network where stub resolver clients query recursive resolvers, or should also apply to the inner parts of the DNS, where recursive resolvers query authoritative servers. In the former case, where most stub resolvers do not perform DNSSEC validation and should not really ask for DNSSEC signatures, then the consequent constraint on DNS response sizes means that IPv6 should be as robust as IPv4 for DNS over UDP.

The authoritative service in this measurement is a dual stack service and uses a 6-node distributed service configuration to minimise latency. It would be reasonable to expect that the name would be resolved on the first query, as the DNS servers were available throughout the 24-hour period. Yet the average query sequence count per unique name to resolve was 2.1 queries. The distribution of query sequence lengths is shown in Figure 1, illustrating that some 46% of query sequences are of length 2 or more. This is somewhat surprising given that the authoritative nameservers are fully available and are not placed under heavy query pressure.

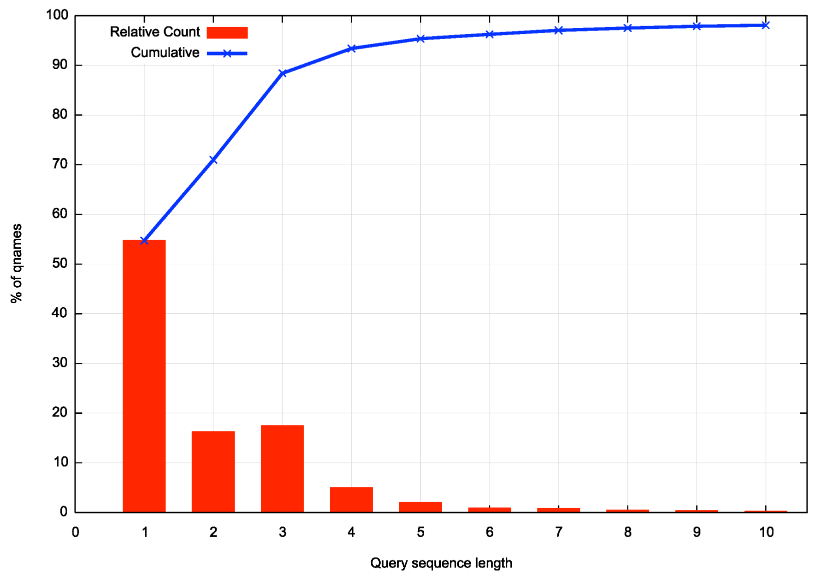

Figure 1 – Distribution of length of query sequences

Of these query sequences, let's now look at the first two queries. The Happy Eyeballs algorithm would suggest that the first DNS query should be performed over IPv6 and the second over IPv4 shortly after. As shown in Table 1, this preference to use IPv6 for the first query, then use IPv4 for the second query occurs in only 22% of the collected query sequences. The major query pattern is to use IPv4 for the first two queries.

First Query	Second Query	Proportion
IPv6	IPv4	22%
IPv4	IPv6	20%
IPv6	IPv6	20%
IPv4	IPv4	39%

Table 1 – First Two Queries

More generally, in looking at all the query sequences of length 2 or greater (16,8091,904 sequences) some 39% of such sequences use IPv4 for all their queries, and 10% exclusively use IPv6.

The Happy Eyeballs approach appears to prefer IPv6for the first query, but to closely follow that initial query with the same query over IPv4, so that if the transaction over IPv6 fails, there is an IPv4 transaction occurring in close succession. Figure 2 looks at the distribution of the elapsed time interval between the first two queries.

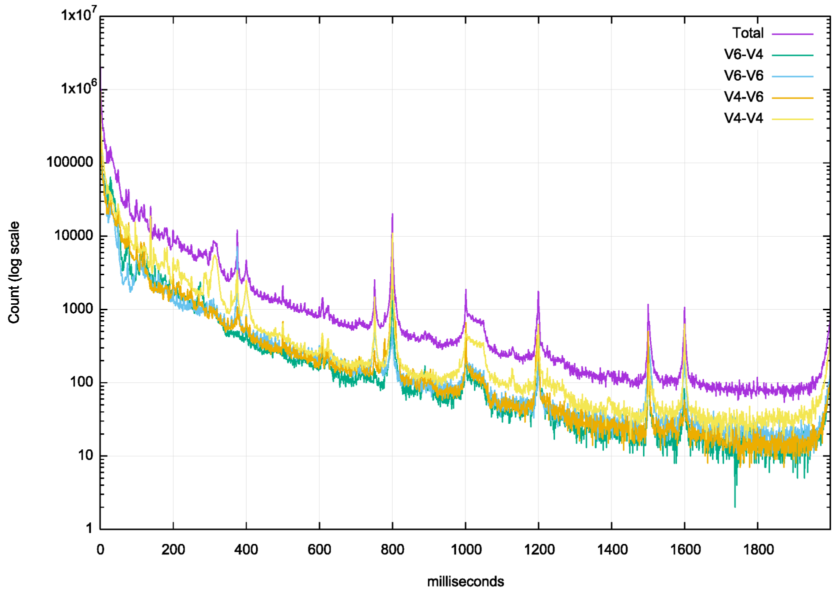

Figure 2 – Distribution of the elapsed time between the first two queries

There are a number of pre-set time intervals between resolver re-queries that are evident here. Some resolvers use a 375ms re-query timer, some use 400ms and some use 1 second. The resolvers' timeout timer value is intended to be long enough to promptly detect response failures while not being too aggressive and triggering unnecessary repeat queries. We can look more closely at the first 20 ms to see if there is any preference for tightly coupling the first two queries, and again no such preference is visible (Figure 3). There is a very slight presence of a 10ms timer.

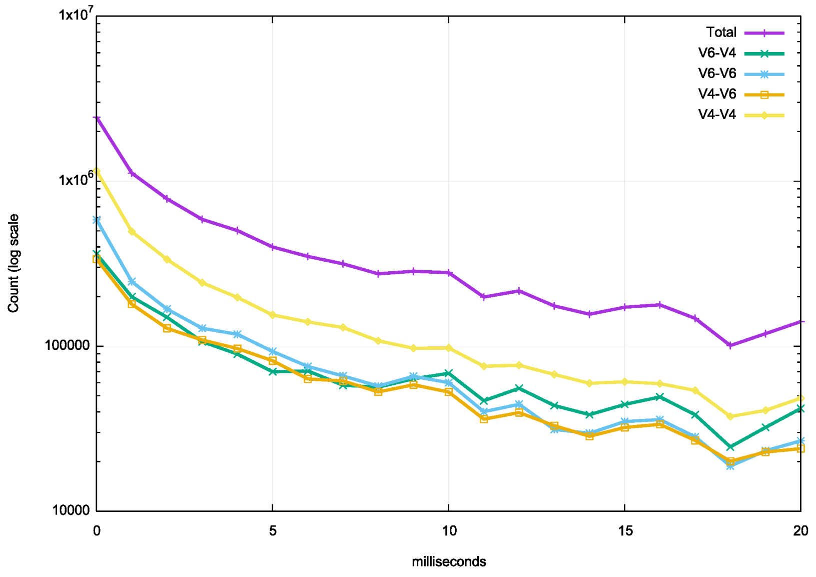

Figure 3 – Distribution of the elapsed time between the first two queries for the first 20ms

It's clear that this recursive resolver / authoritative service DNS environment is aligned to RFC3901 where IPv4 is still relied on for robust behaviour for resolution, particularly for large responses.

DNS Flag Day 2020

While IPv4 can be more robust than IPv6 in coping with large responses over UDP, there are still failure cases. The most common problem is fragment filtering within the network.

The conventional response from DNS resolvers is to re-query using a smaller EDNS buffer size that is unlikely to cause fragmentation, or even query with no buffer size at all, which implies a response must be no larger than 512 bytes. If the response cannot fir within this size, then the responder sets the truncated bit in its response. This is a signal for the client to perform a re-query using TCP.

The EDNS Buffer Size was proposed in RFC 6891 (termed "Payload Size" in this document) to be:

```
A good compromise may be the use of an EDNS maximum payload size of 4096 octets as
a starting point.

A requestor MAY choose to implement a fallback to smaller advertised sizes to work
around firewall or other network limitations. A requestor SHOULD choose to use a
fallback mechanism that begins with a large size, such as 4096.  If that fails, a
fallback around the range of 1280-1410 bytes SHOULD be tried, as it has a
reasonable chance to fit within a single Ethernet frame.  Failing that, a
requestor MAY choose a 512-byte packet, which with large answers may cause a TCP
retry.

   Values of less than 512 bytes MUST be treated as equal to 512 bytes."
```

The issue with this approach was that when fragmented responses are being dropped by the network it takes a timeout interval to conclude that the response has been dropped before re-querying with the smaller buffer size.

The response in 2020 was to propose a "DNS Flag Day" (https://www.dnsflagday.net/2020/) to drop the default EDNS Buffer Size value to 1,232 on all DNS implementations, ensuring that all packets, IPv4 or IPv6 would fit within the minimum MTU used by IPv6. This way the initial response pass through the network unfragmented. If the response was greater than 1,232 bytes then the truncated bit would be set in the response, allowing the client to immediately switch to re-query using TCP. This avoided the client having to wait for a timeput interval.

If all DNS implementations have set this default Buffer Size parameters down to 1,232 the concerns with the operational fragility and the significant IPv6 failure rates would be mitigated and we could once more contemplate a change to RFC3901 to RECOMMEND dual stack servers for authoritative name servers and dual stack recursive resolvers, as in the 3901bis draft. It could even go further and recommend a "IPv6 first" approach to queries, along the lines of Happy Eyeballs, as the operational problems with IPv6 lie in the convoluted and fragile handling of fragmented packets IPv6.

Let's take the same data for one day and look at the offered EDNS Buffer Sizes, as shown in Table 2

Size	Count	%	IPv4	%	IPv6	%
4096	34,047,172	42.3	19,870,478	39.3	14,176,694	47.4
1232	18,934,591	23.5	12,599,895	24.9	6,334,696	21.2
1400	11,645,014	14.5	8,025,462	15.9	3,619,552	12.1
1452	3,908,773	4.9	2,591,661	5.1	1,317,112	4.4
1472	3,029,799	3.8	2,805,058	5.5	224,741	0.8
1220	2,998,080	3.7	1,844,533	3.6	1,153,547	3.9
512	2,167,848	2.7	1,202,667	2.4	965,181	3.2
none	1,118,914	1.4	303,376	0.6	815,538	2.7
1410	732,482	0.9	240,365	0.5	492,117	1.6
1440	595,294	0.7	199,850	0.4	395,444	1.3
1432	273,765	0.3	164,330	0.3	109,435	0.4

Table 2 – Distribution of EDNS Buffer Sizes in measurement queries

The cumulative distribution of these numbers is shown in Figure 4.

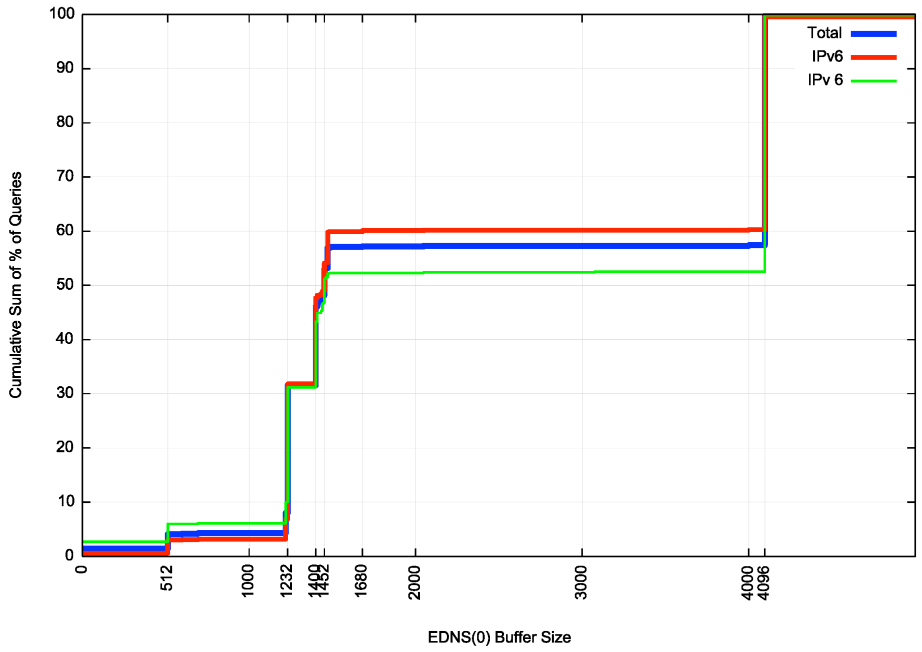

Figure 4 – Cumulative Distribution of ENDS Buffer Sizes used in Recursive to Authoritative Queries

With 43% of observed DNS queries still using an EDNS Buffer Size of 4,096 octets some three years after DNS Flag Day 2020, then I can only conclude that in large parts of the DNS world folk have just stopped listening. Some 47% of IPv6 queries use the old default value of 4,096 bytes, whereas a slightly lower proportion, 39%, use this value in IPv4.

In IPv6 some 69% of queries used an EDNS Buffer Size greater than 1,232, which, when accounting for the overheads of the 8-byte UDP header and the 40-byte IPv6 header, means that just 31% of queries used a buffer size that assuredly avoided DNS fragmentation in the case of IPv6, and with a very high degree of probability in the case of IPv4.

Considering that the de facto MTU of the public Internet is 1,500 octets, then the EDNS Buffer Size value that generally avoids fragmentation is 1,472 in IPv4 and 1,452 in IPv6 (accounting for the 20-byte difference in the IP header size). Some 40% of DNS queries over IPv4 use a buffer size greater than 1,472 and a higher value, namely 49%, of IPv6 queries use a buffer size greater than 1,452.

In the case of IPv4 there is a chance that the network will fragment the packets but a reasonable likelihood that the fragments will get delivered in any case. Using a buffer size above 1,472 is perhaps foolhardy but not completely bonkers!

The same cannot be said about fragmentation and IPv6. The observed failure rate of 41% in DNS resolutions where IPv6 was forced into fragmentation says that fragmentation in IPv6 is just a very time-consuming way of saying to the client: "Better use IPv4!"

What can we do about this?

Writing drafts that proclaim that the DNS SHOULD use IPv6 in such Recursive to Authoritative DNS scenarios is counterproductive in my view, without some further qualification. It simply makes a difficult problem worse for operators who are wanting to advance the state of the IPv6 transition in the DNS.

Rather than meeting the intention of the use of a normative SHOULD, this 3901bis draft as it stands advocates an operational profile that exacerbates an operational problem with IPv6 and packet fragmentation.

What's the answer?

Well, "Don't fragment IPv6 packets" is the obvious answer.

And the way this can be achieved in the DNS with the highest level of assurance is to use an EDNS Buffer Size value of 1,232 in IPv6 DNS configurations. But maybe you might feel that such a setting would be triggering a transition to TCP too early, and a EDNS Buffer Size of 1,452 would achieve the same outcome without unnecessarily moving some queries to TCP.

In any case, maybe the 3901bis draft SHOULD read:

```
In using IPv6 as the platform for DNS queries, DNS implementations SHOULD
use an EDNS Buffer Size value of 1,232 bytes. An operator MAY use a greater
value for this parameter, but only if the DNS operator is confident that
this local setting will not result in IP packet fragmentation being required
to pass a DNS message to its intended recipient.
```

Of course, all DNS software developers, vendors and operators read assiduously study all the DNS RFCs all of the time. So, a reminder that support for TCP is a MUST in the standard DNS specification, as per RFC9210, is unnecessary in this context. Or maybe not!

At that point we might want to consider what form of Happy Eyeballs we might want to adopt for the DNS. A practice of sending an initial query over IPv6 and following this "closely" (i.e., before a response is received to the first query) with the same query over IPv4 represents a piling up of unnecessary query loads onto the DNS. In my view, such a generation of extraneous DNS load through these additional queries falls far short of the objectives of a normative SHOULD. In the worst case it would double the load on the DNS to no major benefit whatsoever. A close coupling of the initial two DNS queries, used in Happy Eyeballs in the application world, is a poor model to follow for the DNS itself. If we are using EDNS Buffer Size values that raise the level of assurance that a DNS response will be received to all queries, then perhaps a simpler form of protocol management is sufficient, namely that:

```
If the reduced EDNS Buffer Size parameter is used by a DNS resolver, then
such DNS resolvers MAY order the list of servers that could be queried to
prefer to use an IPv6 query as the initial query.
```

And this simple measure would be entirely sufficient for the DNS to support IPv6 in the context of the broader IPv6 transition within the overall Internet.

Disclaimer

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

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