DNS over IPv6 – A Study in Fragmentation

DNS OARC, September 2017

Geoff Huston, Joao Damas

APNIC Labs

What happens to a "large" DNS response?

(Where “large” is > 1280 octets)

What happens to a "large" DNS response?

(Where “large” is > 1280 octets)

For example:

- What happens in a signed DNSSEC response when there are multiple RRSIG records?

What happens to a "large" DNS response?

```
$ dig +dnssec DNSKEY org
; <<>> DiG 9.8.3-P1 <<>> +dnssec DNSKEY org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21353
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512;
; QUESTION SECTION:
;org.       IN          DNSKEY

;; ANSWER SECTION:
org.        861         IN          DNSKEY      256 3 7 AwEAAXxsMmN/JgpEE9Y4uFNRJm7Q9GBwmEYUCsCxuKlg
org.        861         IN          DNSKEY      256 3 7 AwEAAayiVbuM+ehlsKsuAL1CI3mA+5JM7ti3VeY8ysmo
org.        861         IN          DNSKEY      257 3 7 AwEAAZTjbIO5kIpxWUtyXc8avsKyHIIZ+LjC2Dv8naO+
org.        861         IN          DNSKEY      257 3 7 AwEAAcMnWBKLuvG/LwnPVykcmpvnntwxfshHlHRhlY0F

org.        861         IN          RRSIG       DNSKEY 7 1 900 20170815152632 20170725142632 3947
org.        861         IN          RRSIG       DNSKEY 7 1 900 20170815152632 20170725142632 9795
org.        861         IN          RRSIG       DNSKEY 7 1 900 20170815152632 20170725142632 17883

;; Query time: 134 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jul 31 12:07:16 2017
;; MSG SIZE  rcvd: 1625
```

The response to a DNSKEY query for .org uses a response of 1,625 octets!

What happens to a "large" DNS response?

(Where “large” is > 1280 octets)

For example:

- The case of the upcoming key roll, when the response to a DNSKEY query will be 1,414 octets for a few weeks (and 1,425 octets at the end of the roll)?

What we expect…

When a resolver offers **no** EDNS(0) UDP buffer size then the server offers a truncated UDP response no larger than 512 octet of DNS payload

> The resolver should be capable of interpreting this truncated response as a signal to re-query using TCP

What we expect…

When resolvers offer a large UDP buffer size in the EDNS(0) query options then this denotes a capability of the resolver to process a large response – the server may then send a large UDP response which may involve UDP fragmentation

The implicit assumption in the query's EDNS(0) buffer size offer is that the network path is equally capable of handling large UDP responses

However…

UDP Fragmentation has its problems

- UDP trailing fragments in IPv4 and IPv6 may encounter fragment filtering rules on firewalls in front of resolvers

However…

UDP Fragmentation has its problems

- UDP trailing fragments in IPv4 and IPv6 may encounter fragment filtering rules on firewalls in front of resolvers

- Large UDP packets in IPv6 may encounter path MTU mismatch problems, and the ICMP6 Packet Too Big diagnostic message may be filtered.

  Even if it is delivered, the host may not process the message due to the lack of verification of the authenticity of the ICMP6 message. Because the protocol is UDP, receipt of an ICMP6 message will not cause retransmission of a re-framed packet.

However…

UDP Fragmentation has its problems

- UDP trailing fragments in IPv4 and IPv6 may encounter fragment filtering rules on firewalls in front of resolvers

- Large UDP packets in IPv6 may encounter path MTU mismatch problems, and the ICMP6 Packet Too Big diagnostic message may be filtered.

  Even if it is delivered, the host may not process the message due to the lack of verification of the authenticity of the ICMP6 message. Because the protocol is UDP, receipt of an ICMP6 message will not cause retransmission of a re-framed packet.

- UDP fragments in IPv6 are implemented by Extension Headers. There is some evidence of deployment of IPv6 switching equipment that unilaterally discards IPv6 packets with extension headers (RFC 7872)

For example:

The only authoritative name server for this zone is via iPv6

```
$ dig +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8

; <<>> DiG 9.9.5-9+deb8u10-Debian <<>> +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 34058
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1


;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;question.ap2.dotnxdomain.net. IN    A


;; Query time: 3477 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 06 04:57:41 UTC 2017
;; MSG SIZE rcvd: 104
```

Yes, i'm asking Google's public DNS resolver service over iPv4, and getting Googles PDNS to query the authoritative server over iPv6

For example:

The only authoritative name server for this zone is via iPv6

```
$ dig +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8

; <<>> DiG 9.9.5-9+deb8u10-Debian <<>> +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 34058
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
```

Oops!

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;question.ap2.dotnxdomain.net. IN    A

;; Query time: 3477 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 06 04:57:41 UTC 2017
;; MSG SIZE rcvd: 104
```

Yes, i'm asking Google's public DNS resolver service over iPv4, and getting Googles PDNS to query the authoritative server over iPv6

For example:

The only authoritative name server for this zone is via IPv6

```
$ dig +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8

; <<>> DiG 9.9.5-9+deb8u10-Debian <<>> +bufsize=4096 +dnssec question.dotnxdomain.net. @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 34[illegible]
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1


;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;question.ap2.dotnxdomain.net. IN A



;; Query time: 3477 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 06 04:57:41 UTC 2017
;; MSG SIZE rcvd: 104
```

if the IPv6-only response from the authoritative name server to Google's name server exceeds 1500 octets the Google resolver returns SERVFAIL, despite providing a large UDP Buffer size option to the authoritative name server

Yes, I'm asking Google's public DNS resolver service over IPv4, and getting Googles PDNS to query the authoritative server over IPv6

However…

- UDP Fragmentation has its problems
  - UDP trailing fragments in IPv4 and IPv6 may encounter fragment filtering rules on firewalls in front of resolvers
  - Large UDP packets in IPv6 may encounter path MTU mismatch problems, and the ICMP6 Packet Too Big diagnostic message may be filtered.

    Even if it is delivered, the host may not process the message due to the lack of verification of the authenticity of the ICMP6 message. Because the protocol is UDP, receipt of an ICMP6 message will not cause retransmission of a re-framed packet.
  - UDP fragments in IPv6 are implemented by Extension Headers. There is some evidence of deployment of IPv6 switching equipment that unilaterally discards IPv6 packets with extension headers (RFC 7872)
- DNS over TCP is not always available
  - TCP over DNS may be blocked by firewall filtering rules even when the resolver re-queries using a smaller EDNS0 buffer size to generate a truncated response (*)

* http://www.potaroo.net/presentations/2013-10-05-dns-protocol.pdf
DNS OARC, October 2013

TCP may not help here

- If the fragmented response is lost (firewalls having issues with trailing fragments, PMTU signal loss, or network switches discarding IPv6 packets with Extension Headers) then the client gets no response at all, which is a signal of server failure
- A non-responsive server may prompt a client to repeat the UDP query, which won’t help in this case.
- This repeated unresponsive behaviour is not necessarily an invitation to re-query using TCP unless the client persists and re-issues the query with a small (or no) EDNS(0) UDP Buffer size option

How can we measure this scenario of large DNS responses, DNS resolvers and IPv6?

Our measurement approach

We use the Ad platform to enroll endpoints to attempt to resolve a set of DNS names:

- Each endpoint is provided with a unique name string (to eliminate the effects of DNS caching)
- The DNS name is served from our authoritative servers
- Each DNS name contains a name creation time component (so that we can disambiguate subsequent replay from original queries)

Experiment Parameters

Three DNS response sizes:

- Small – 169 octet IPv6 packet response
- Medium – 1,428 octet IPv6 packet response
- Large – 1,886 octet IPv6 payload

Operate the experiment's servers in V6 only, using a 1,280 octet MTU on the DNS server

If the client's DNS resolvers can successfully resolve the DNS name they they will fetch the named web object, so the measurement here is one of the failure rate to access the web object

Dual Stack Resolvers

Not every resolver can perform a DNS query using IPv6

48% of the 65M experiments could **not** use the DNS over IPv6 to query for the name server record

IPv6 DNS Resolution Results

48% of the 65M experiments could **not** use the DNS over IPv6 to query for the name server record

Of those that did, we observed the following Loss Rates in fetching the web object:

SMALL: **7%**

MEDIUM: **42%**

LARGE: **42%**

Who, where why?

Are there regional differences?

We use three different servers, and divide (roughly) clients into three geo areas: Americas, Europe & Africa, Asia & Oceania

LOSS RATE	Americas	Europe	Asia
SMALL	4%	5%	13%
MEDIUM	23%	49%	52%
LARGE	24%	50%	52%

V6, the DNS and Large Responses

A V6 only-Authoritative DNS Server serving UDP fragmented DNS responses appears to have significant problems in delivering this UDP fragmented response to recursive resolvers in the Internet

However the 7% loss rate for the unfragmented DNS response points to a high noise rate in the data collected using this experimental technique

Can we identify these resolvers?

This is a challenging problem:

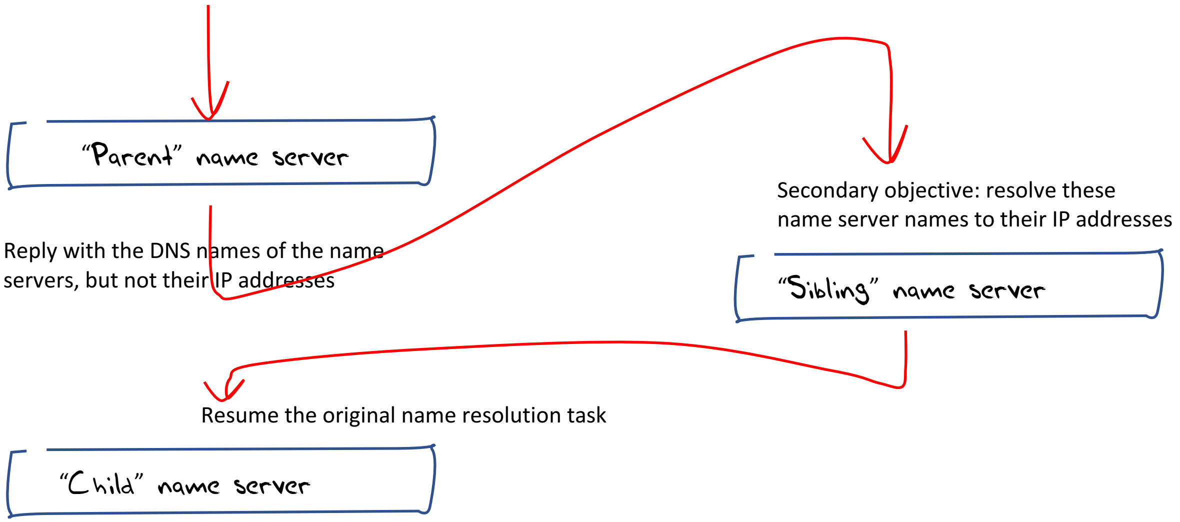
- An Extension Header packet drop is a silent drop
  - no ICMP6 notification back to the sender
- The recursive resolver will receive no answer to its query, and will be unable to answer back to the stub resolver. However it may elect to re-query the authoritative server before giving up
- A stub DNS resolver client will time out on the query to a recursive resolver, and may re-query to other recursive resolvers before giving up

Second Experiment

We created a ”glueless” delegation in the DNS

- The response to the query to the ‘parent’ lists the name servers of the ‘child’, but deliberately withholds the IP address of these name servers in the response. i.e. the response is missing the ‘glue’ records for the name servers
  - We do this on the fly to ensure that each query name is unique, to ensure that there is no unintended side effect from DNS caching
- We then inflated the response of the name server record by adding pad records to the response
- The idea is that the name will only be resolved if the resolver is capable of receiving a large response when trying to chase down the name server addresses

"Glueless" Delegation



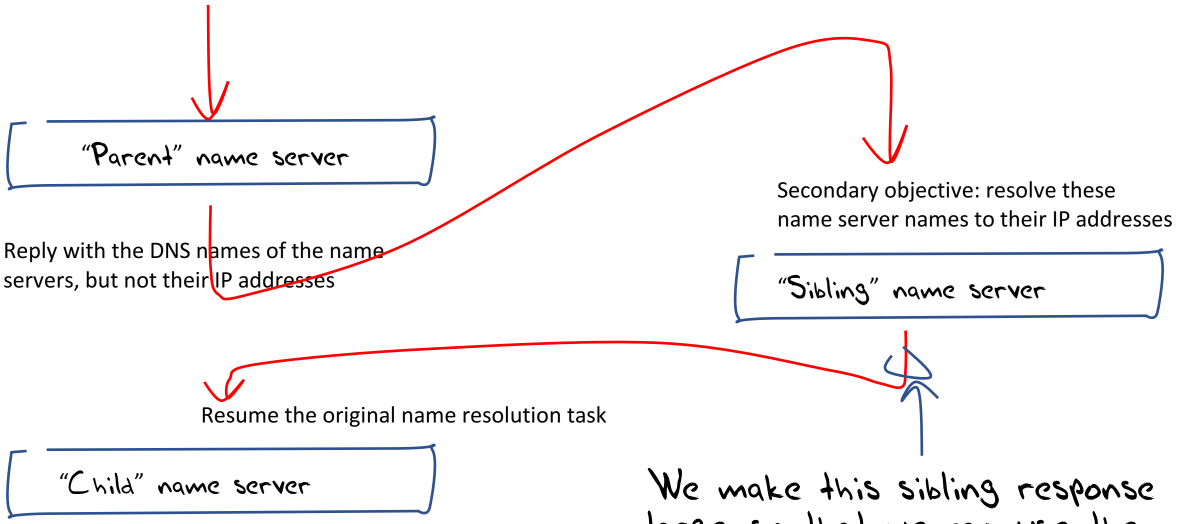
Reply with the DNS names of the name servers, but not their IP addresses

Secondary objective: resolve these name server names to their IP addresses

Resume the original name resolution task

The "child" name server will only be queried if the resolver could receive the response from the sibling name server

"Glueless" Delegation



Reply with the DNS names of the name servers, but not their IP addresses

Secondary objective: resolve these name server names to their IP addresses

Resume the original name resolution task

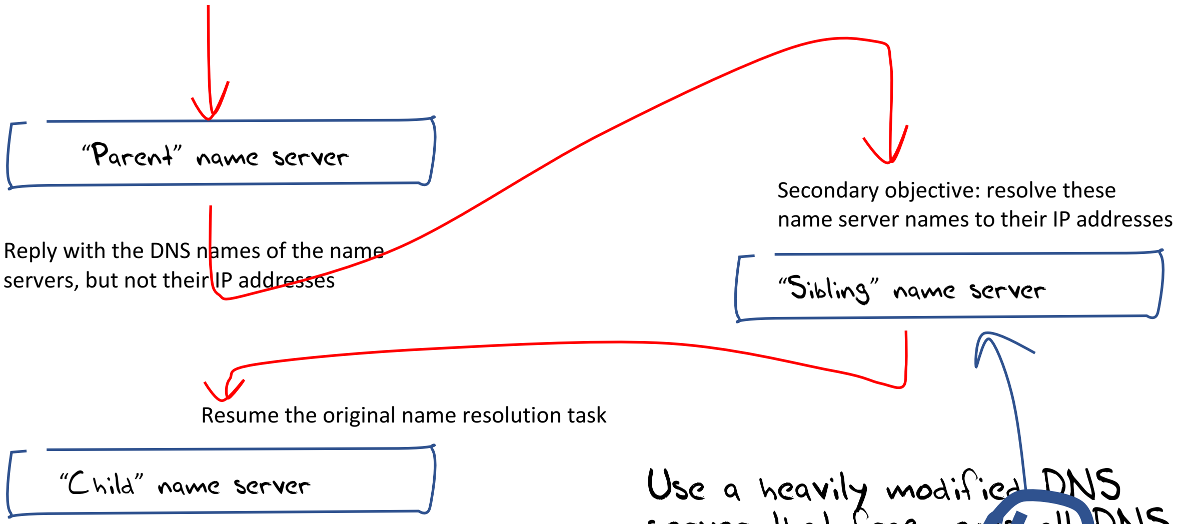
We make this sibling response large, so that we can use the queries at the child to determine if the resolver can receive these large responses?

The "child" name server will only be queried if the resolver could receive the response from the sibling name server

Fail!

- Many resolvers deliberately omit EDNS(0) options in their queries to chase down name server addresses
  - As these records are not part of a subsequent DNSSEC validation pass then any DNSSEC records, such as RRSIG records are not needed, and most responses will be under 512 bytes in any case
  - And those that are not under 512 will be truncated, allowing the resolver to re-ask the query using TCP
- But we wanted to pass the resolver a large fragmented UDP response to see if received it
- If we can’t do “*large fragmented*” can we just do “*fragmented*”?

"Glueless" Delegation and Gratuitous IPv6 Fragmentation



"Parent" name server

Reply with the DNS names of the name servers, but not their IP addresses

Secondary objective: resolve these name server names to their IP addresses

"Sibling" name server

Resume the original name resolution task

"Child" name server

The "child" name server will only be queried if the resolver could receive the response from the sibling name server

Use a heavily modified DNS server that fragments all DNS responses, irrespective of their size!

V6, the DNS and Fragmented UDP Responses

We used the Ad platform to enroll endpoints to attempt to resolve a DNS name that included a IPv6 fragmented UDP response when attempting to resolve the name server's name

Total number of tests: 10,851,323

Failure Rate in receiving a large response: 4,064,356

IPv6 Fragmentation Failure Rate: **38%**

Where?

Regionally there are slightly different outcomes:

Asia Pacific:	31% Failure
Americas:	37% Failure
Eurasia & Africa:	47% Failure

Which Resolvers?

- 10,115 IPv6 seen resolvers
- 3,592 resolvers were consistently unable to resolve the target name (likely due to failure to receive the fragmented response)
- Which is too large a list to display here
- But we can show the top 20…

Individual IPv6 Resolvers in Which Networks?

Resolver	Hits	AS	AS Name	CC
2405:200:1606:672::5	4,178,119	55836	RELIANCEJIO-IN Reliance Jio Infocomm Limited	IN
2402:8100:c::8	1,352,024	55644	IDEANET1-IN Idea Cellular Limited	IN
2402:8100:c::7	1,238,764	55644	IDEANET1-IN Idea Cellular Limited	IN
2407:0:0:2b::5	938,584	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2a::3	936,883	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2a::6	885,322	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2b::6	882,687	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2b::2	882,305	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2a::4	881,604	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2a::5	880,870	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2a::2	877,329	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2b::4	876,723	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:2b::3	876,150	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2402:8100:d::8	616,037	55644	IDEANET1-IN Idea Cellular Limited	IN
2402:8100:d::7	426,648	55644	IDEANET1-IN Idea Cellular Limited	IN
2407:0:0:9::2	417,184	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:8::2	415,375	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:8::4	414,410	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:9::4	414,226	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
2407:0:0:9::6	411,993	4761	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID

All these resolvers appears to be unable to receive fragmented UDP DNS responses – This is the Top 20, as measured by the query count per resolver address

Resolver 'Farms' in Which Networks?

AS	Hits	% of Total	AS Name	CC
15169	7,952,272	17.3%	GOOGLE - Google Inc.	US
4761	6,521,674	14.2%	INDOSAT-INP-AP INDOSAT Internet Network Provider	ID
55644	4,313,225	9.4%	IDEANET1-IN Idea Cellular Limited	IN
22394	4,217,285	9.2%	CELLCO - Cellco Partnership DBA Verizon Wireless	US
55836	4,179,921	9.1%	RELIANCEJIO-IN Reliance Jio Infocomm Limited	IN
10507	2,939,364	6.4%	SPCS - Sprint Personal Communications Systems	US
5650	2,005,583	4.4%	FRONTIER-FRTR - Frontier Communications of America	US
2516	1,322,228	2.9%	KDDI KDDI CORPORATION	JP
6128	1,275,278	2.8%	CABLE-NET-1 - Cablevision Systems Corp.	US
32934	1,128,751	2.5%	FACEBOOK - Facebook	US
20115	984,165	2.1%	CHARTER-NET-HKY-NC - Charter Communications	US
9498	779,603	1.7%	BBIL-AP BHARTI Airtel Ltd.	IN
20057	438,137	1.0%	ATT-MOBILITY-LLC-AS20057 - AT&T Mobility LLC	US
17813	398,404	0.9%	MTNL-AP Mahanagar Telephone Nigam Ltd.	IN
2527	397,832	0.9%	SO-NET So-net Entertainment Corporation	JP
45458	276,963	0.6%	SBN-AWN-AS-02-AP SBN-ISP/AWN-ISP and SBN-NIX/AWN-NIX	TH
6167	263,583	0.6%	CELLCO-PART - Cellco Partnership DBA Verizon Wireless	US
8708	255,958	0.6%	RCS-RDS 73-75 Dr. Staicovici	RO
38091	255,930	0.6%	HELLONET-AS-KR CJ-HELLOVISION	KR
18101	168,164	0.4%	Reliance Communications DAKC MUMBAI	IN

This is the total per origin AS of those resolvers that appear to be unable to receive fragmented UDP DNS responses. This is the Top 20, as measured by the query count per origin AS

What's the Problem?

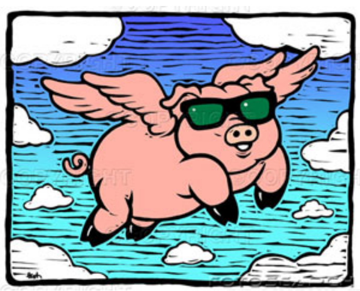
Extension Headers require that any transport protocol-sensitive functions in switches need to unravel the packet header's extension header chain

- This takes a variable number of cycles for the device
- This is an anathema to a switch
- And passing through extension headers that the switch does not understand or not prepared to check is a security risk
- Easier to drop all packets with extension headers!

See RFC 7872

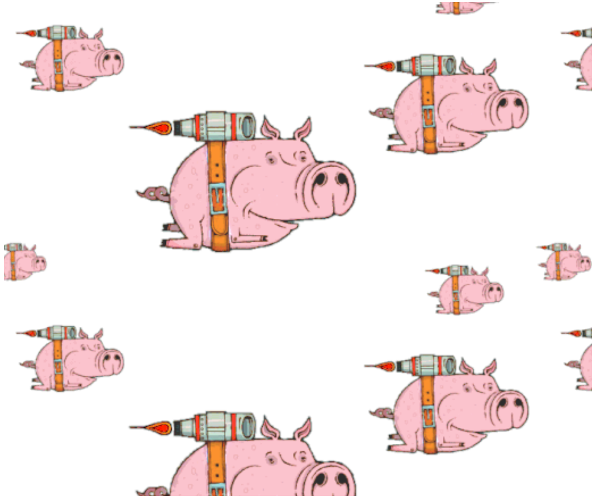
What can we do about it?

A. Get all the deployed routers and switches to accept packets with IPv6 Fragmentation Headers



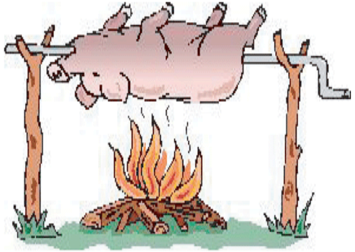
What can we do about it?

B. Alter the way IPv6 describes packet fragmentation



What can we do about it?

C. Move the DNS off UDP



Which Means…

We can try and ensure that there will always be enough IPv4 around to keep the DNS running over UDP long after all other traces of IPv4 in the Internet have been extinguished

Or we could try and constrain the bloat pressure in the DNS to keep responses from pushing into UDP fragmentation

Or think about moving to a different approach to the carriage protocol for the DNS in an IPv6 environment

Seriously

- If we are going to take a future of an IPv6-only Internet seriously we are going to have to take the concept of a DNS over IPv6 equally seriously
- Which means that we need to figure out how to change the appalling drop rate for fragmented UDP responses for DNS over IPv6 in today's network

Some current ideas

Change the protocol behaviour?

- Shift Additional Records into additional explicit UDP query/response transactions rather than bloating the original DNS response
- ATR: Add a truncated response to trail a fragmented response

Change the transport:

- DNS over TCP by default?
- DNS over TLS over TCP by default?
- DNS over QUIC?
- Devise some new DNS framing protocol that uses multiple packets?

Acknowledgements and Thanks

- Google has supported our efforts to conduct this measurement since its inception, and continues to support us.
- ICANN provide support for this measurement activity, particularly relating to our measurement of DNS and DNSSEC capabilities
- ISC for a conveniently located server at PAIX
- Fernando Gont’s IPv6 toolbox provided valuable hints on the raw IPv6 packet handling routines we developed to perform IPv6 fragmentation handling tests
- And thanks to Ray Bellis for development of the original dynamic DNS server code

Thanks!