Surviving IPv6 Fragmentation

Geoff Huston
APNIC Labs

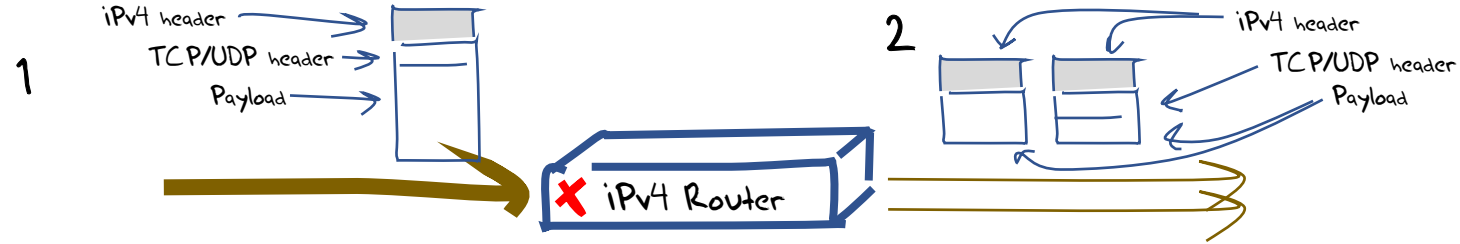
IPv6 and Packet Fragmentation

IPv6 made two major changes to IP's handling of packet fragmentation:

- The fragmentation control header has been moved out of the IP header to become an **extension header**
  - In other words the UDP / TCP protocol header is pushed further into the packet and to find it you need to follow the header chain
- The IPv4 'Don't Fragment' bit is jammed **on**
  - In the case of path MTU issues IPv6 routers should not perform fragmentation on the fly, but are required to pass an ICMPv6 PTB message back to the packet's sender

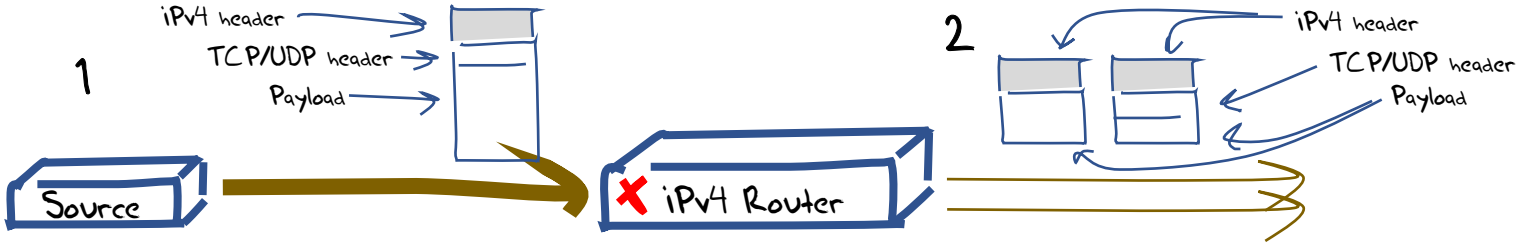
IPv4 and IPv6 handling of Path MTU issues

IPv4 "Forward Fragmentation"

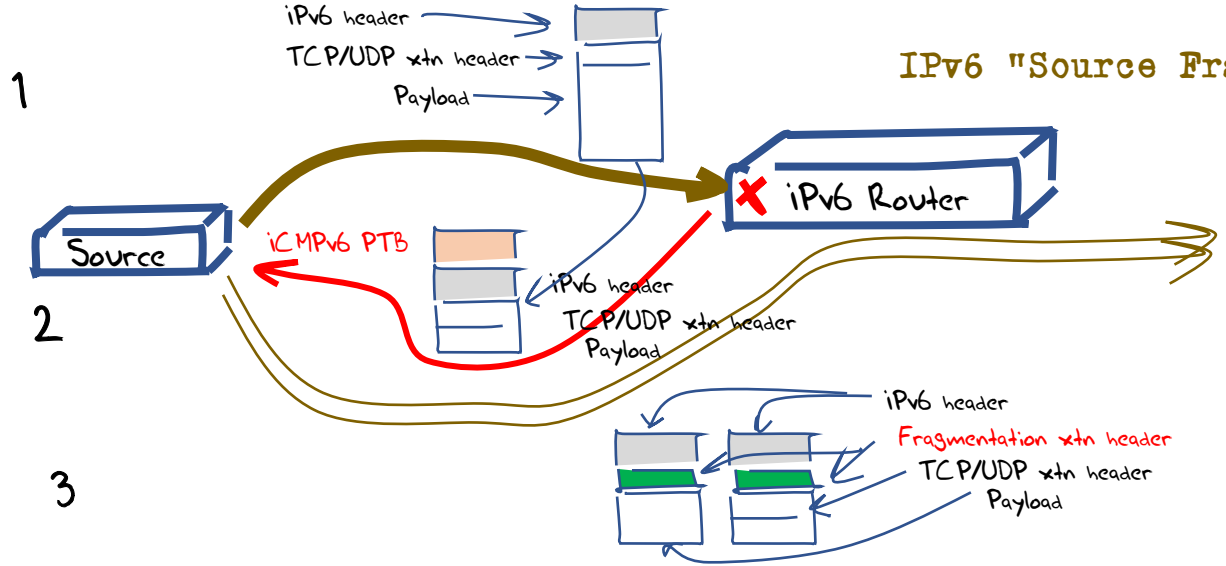


IPv4 and IPv6 handling of Path MTU issues

IPv4 "Forward Fragmentation"



IPv6 "Source Fragmentation"



New Dependencies

For IP fragmentation to work in IPv6 then:

- all ICMPv6 messages have to be passed **backwards** from the interior of the network to the sender

and

- IPv6 packets containing a IPv6 Fragmentation Extension header should **not** be dropped

ICMPv6 PTB handling

Only the sending host now has control of fragmentation

A received ICMPv6 message needs to alter the sender's state to that destination:

> For TCP, if the ICMP payload contains the TCP header, then you can pass this to the TCP control block. TCP can alter the session MSS and resend the dropped data, or you can just alter the local per-destination MSS and hope that TCP will be prompted to resend

ICMPv6 PTB handling

Only the sending host now has control of fragmentation

A received ICMPv6 message needs to alter the sender's state to that destination:

For UDP – um, err, um well

ICMPv6 PTB handling

Only the sending host now has control of fragmentation

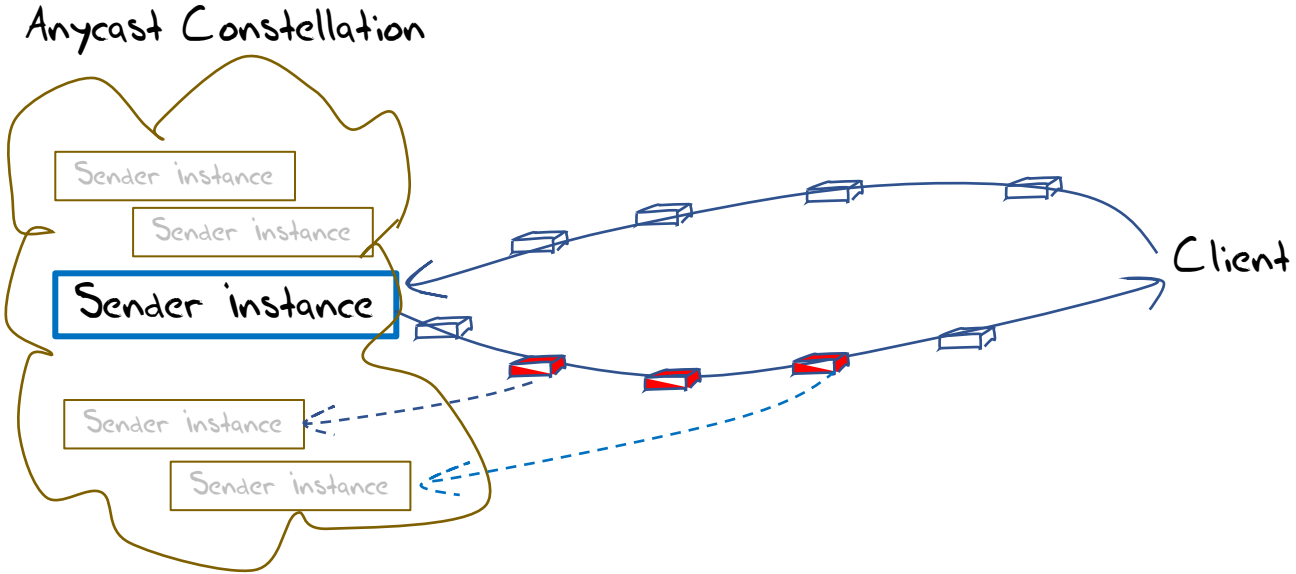
A received ICMPv6 message needs to alter the sender's state to that destination:

For UDP – um, err, um well

Maybe you should store the revised path MTU in a per-host forwarding table cache for a while

If you ever need to send another UDP packet to this host you can use this cache entry to guide your fragmentation behaviour

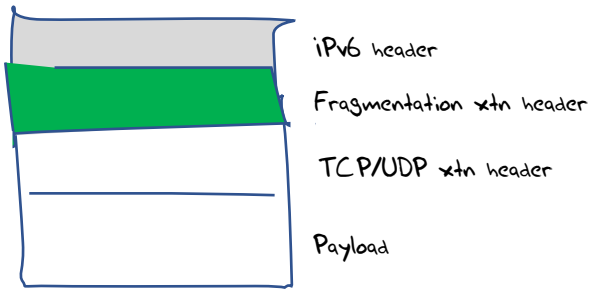
ICMPv6 PTB and Anycast



It is not obvious (or even assured) that every router on the path from an anycast instance to a client host will necessarily be part of the same anycast instance “cloud”

The implication is that in anycast, the reverse ICMPv6 PTB messages will not necessarily head back to the original sender!

IPv6 Fragmentation Extension Header



The extension header sits between the IPv6 packet header and the upper level protocol header for the leading fragged packet, and sits between the header and the trailing payload frags for the trailing packets

Practically, this means that transport-protocol aware packet processors/switches need to decode the extension header chain, if its present, which can consume additional cycles to process/switch a packet – and the additional time is not predictable. For trailing frags there is no transport header!

Or the unit can simply discard all IPv6 packets that contain extension headers - which is what a lot of transport protocol sensitive IPv6 deployed switching equipment appears to do!

Who uses Fragmentation anyway?

- Well, the DNS is a good place to start looking!

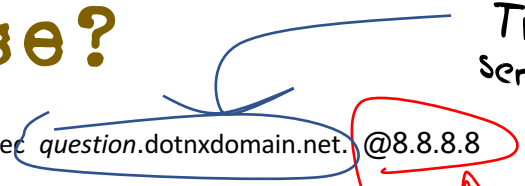
Who uses Fragmentation anyway?

- Well, t

```
$ 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 can happen to a "large" DNS response?



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
```

Oops!

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

What can happen to a "large" DNS response?

```
$ 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: 3477
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

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

;; Query time: 8477 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 06 04:57:41 UTC 2017
;; MSG SIZE  rcvd: 104
```

The only authoritative name server for this zone is via IPv6

If the IPv6-only response from the authoritative name server exceeds 1500 octets the Google resolver Google's name server 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

The DNS expects Fragmentation to just work!

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

When a resolver offers a large EDNS(0) UDP buffer size 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

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

Is this a problem for today's IPv6 Internet?

- Can we measure the extent to which users might be affected with this scenario of large DNS responses, DNS resolvers and IPv6?

IPv6 Fragmentation Handling

There is a lot of “drop” behaviour in the Ipv6 Internet for Fragmentation Extension headers

RFC7872 – recorded EH packet drop rates of 30% - 40%

This experiment sent fragmented packets towards well-known servers and observed whether the server received and reconstructed the fragmented packet

But sending fragmented queries to servers is not all that common – the reverse situation of big responses is more common

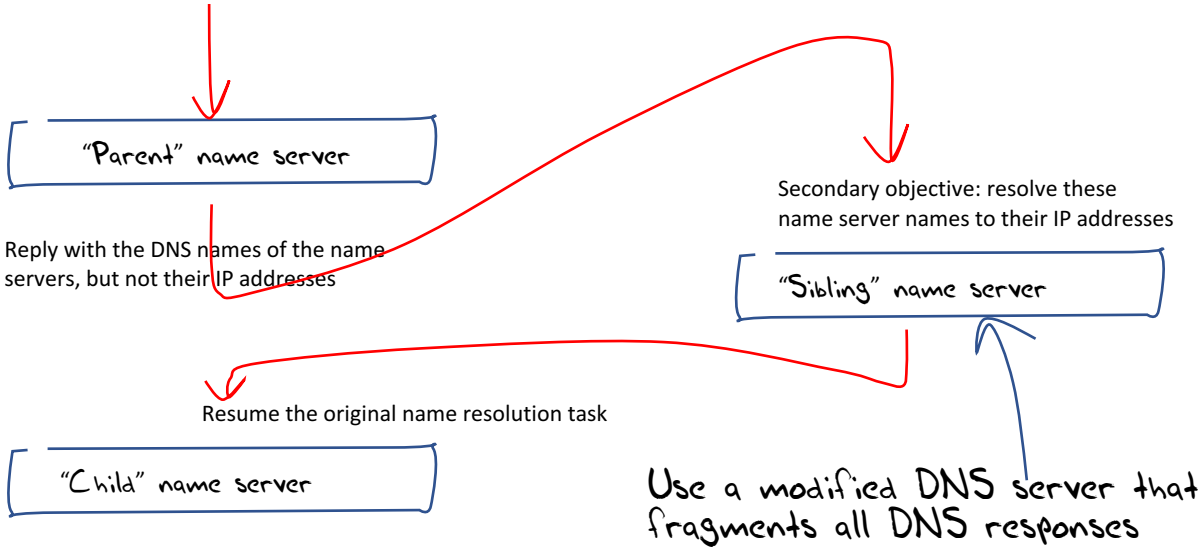
So what about sending fragmented packets BACK from servers – what’s the drop rate of the reverse case?

Our Measurement Approach

We use an Online 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
- Resolving the DNS name requires the user's DNS resolvers to receive a fragmented IPv6 packet

"Glueless" Delegation to detect IPv6 Fragmentation Handling



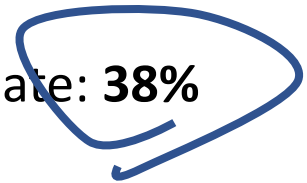
The "child" name server will only be queried if the resolver could receive the response from the sibling name server

V6, the DNS and Fragmented UDP

Total number of tests: 10,851,323

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

IPv6 Fragmentation Failure Rate: **38%**



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…

Which Resolvers?

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

Resolvers 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 about TCP and Fragmentation?

Let’s try the same approach:

- Set up an ad-based measurement using a customised IPv6 packet handler
- Pass all TCP responses through a packet fragmenter
- Use a packet capture to see if the fragmented TCP segment was ACKed or not

What about TCP and Fragmentation?

1,961,561 distinct IPv6 end point addresses

434,971 failed to receive Fragmented IPv6 packets

22% failure rate

Where are TCP e-2-e drops?

AS	Samples	Failure Rate	AS Name	CC
3598	4,762	99.4%	MICROSOFT-CORP-AS - Microsoft Corporation	US
15169	6,426	98.9%	GOOGLE - Google Inc.	US
24961	252	98.4%	MYLOC-AS	DE
6621	4,431	92.8%	HNS-DIRECPC - Hughes Network Systems	US
131222	595	89.1%	MTS-INDIA-IN 334, Udyog, Vihar	IN
38229	260	86.5%	LEARN-LK Lanka Education & Research Network	LK
6939	106,057	85.2%	HURRICANE - Hurricane Electric	US
852	4,552	84.1%	ASN852 - TELUS Communications Inc.	CA
32934	359	79.7%	FACEBOOK - Facebook	US
54115	128	78.9%	FACEBOOK-CORP - Facebook Inc	US
1312	122	76.2%	Virginia Polytechnic Institute and State Univ.	US
22394	109,333	73.2%	CELLCO - Cellco Partnership DBA Verizon Wireless	US
5603	1,938	69.3%	SIOL-NET	SI
4134	171	69.0%	CHINANET-BACKBONE No.31	CN
20845	272	68.4%	DIGICABLE	HU

Top 15 networks with highest Fragmented IPv6 Drop Rates

Why do we see these high packet drop rates?

Two major factors appear to lie behind this failure rate:

- Network equipment dropping IPv6 packets with Extension Headers
- Firewalls dropping Fragmented packets

Why do we see these high packet drop rates?

Two major factors appear to lie behind this

- Network

the underlying large packet response failure rate could be higher than the numbers presented here – our experiment gratuitously fragmented the IPv6 packet and did not rely on receipt of an ICMP6 PTB message to trigger this action – so these numbers don't factor in the potential for additional failures caused by ICMP6 PTB message filtering

What to do?

Accepting a future IPv6-only Internet means we are going to have to take the problem of IPv6 Fragmentation seriously

- Because relying on IPv4 as a backup is a hack with an indeterminate future!

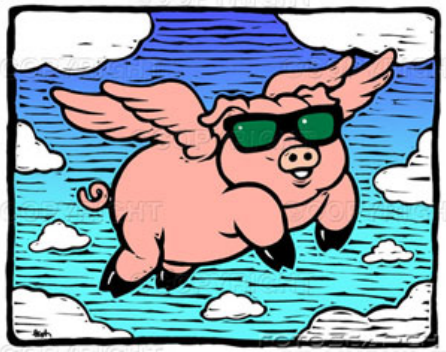
Which means that we need to figure out how to change the appalling drop rate for fragmented IPv6 packets both in the DNS and in end-to-end paths in the net

Should we try and fix the network problem or try to work around it?

What can we do about it?

Fix it!

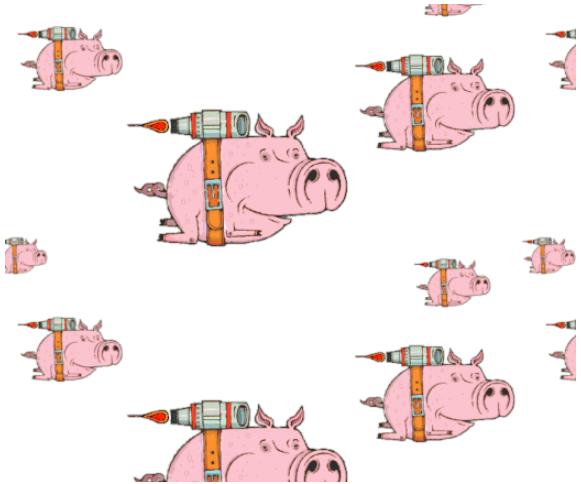
Get all the deployed routers, switches and firewalls and related network middleware to accept packets with IPv6 Fragmentation Headers



What can we do about it?

Change it!

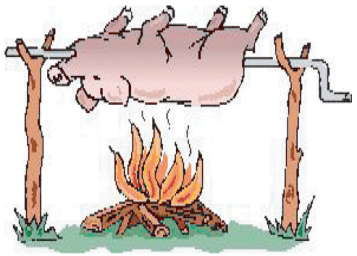
Change the way in which IPv6 manages IP fragmentation and the use of Extension Headers as Fragmentation Control fields



What can we do about it?

Avoid it!

Change application behaviour so as to avoid the use of packet fragmentation completely



Pick one?

All of these options have a certain level of pain, cost and potential inconvenience

Its hard to work out what is the best course of action, but it seems like a lot of extra effort if we take on all three at once!

For TCP

Working around this issue in TCP can be as simple as a very careful selection of a default IPv6 TCP MSS

- Large enough enough to offer a tolerable data carriage efficiency
- Small enough to avoid Path MTU issues

And perhaps you might want to to support TCP path MTU discovery (RFC 4281)

For UDP

- Working around this issue can be challenging with UDP
  - ICMPv6 PTB filtering causes silence
  - Fragment drop is silent

- An effort to work around this necessarily involves application-level adaptation to pass large responses without relying on UDP packet fragmentation

Large DNS Responses and IPv6

Change the protocol behaviour?

- Shift Additional Records into additional explicit UDP query/response transactions rather than bloating the original DNS response
- Perform UDP MTU discovery using EDNS(0) UDP Buffer Size variations as a probe
- Add a truncated minimal UDP response to trail a fragmented response (ATR)

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 instead of fragmentation

Where now?

- We have a decent idea of the problem space we need to resolve
- We'd prefer a plan that allows each of us to work independently rather than a large scale orchestrated common change
- We're not sure we can clean up all the ICMPv6 filters and EH packet droppers in the IPv6 network
- And it sure seems a bit late in the day to contemplate IPv6 protocol changes
- Which means that we are probably looking at working around the problem by changing the behaviour of applications

What do the RFC's say?

What do the RFC's say?

Internet Engineering Task Force (IETF)
Request for Comments: 8085
BCP: 145
Obsoletes: 5405
Category: Best Current Practice
ISSN: 2070-1721

L. Eggert
NetApp
G. Fairhurst
University of Aberdeen
G. Shepherd
Cisco Systems

March 2017

UDP Usage Guidelines

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. This document provides guidelines on the use of UDP for the designers of applications, tunnels, and other protocols that use UDP. Congestion control guidelines are a primary focus, but the document also provides guidance on other topics, including message sizes, reliability, checksums, middlebox traversal, the use of Explicit Congestion Notification (ECN), Differentiated Services Code Points

What do the RFC's say?

Internet Engineering Task Force (IETF)
Request for Comments: 8085
BCP: 145
Obsoletes
Category:
ISSN: 207

L. Eggert
NetApp

Abstract

The Us
transp
docume
applic
contro
provid
reliab
Conges

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191] [RFC1981] [RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation.

However, the ICMP messages that enable path MTU discovery are being increasingly filtered by middleboxes (including Firewalls) [RFC4890]. When the path includes a tunnel, some devices acting as a tunnel ingress discard ICMP messages that originate from network devices over which the tunnel passes, preventing these from reaching the UDP endpoint.

What do the RFC's say?

Internet Engineering Task Force (IETF)
Request for Comments: 8085
BCP: 145
Obsoletes
Category:
ISSN: 207

L. Eggert
NetApp

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191]

Abstract

The Us
transp
docume
applic
contro
provid
reliab
Conges

Applications that do not follow the recommendation to do PMTU/PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460].

endpoint.

What do the RFC's say?

Internet Engineering Task Force (IETF)
Request for Comments: 8085
BCP: 145
Obsoletes
Category:
ISSN: 207

L. Eggert
NetApp

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmiss t (MTU) along the path to the destination. Conse application SHOULD either use the path

Abstr

Applications that
disco PMTUD
in sult
is MTU
tha s
in _S
firs and the
1280 bytes [RFC2460].

This BCP is saying that using EDNS(0) in the DNS to signal the capability of accepting large fragmented DNS responses was unwise, and if a host/application does know the path MTU, it should truncate at UDP at 1280 octets (and IPv4 should truncate at 512 octets!)

Where are we?

- Is the root cause problem with the way our IPv6 networks handle Fragmented IPv6 packets?
- Or with the way our IPv6 networks handle IPv6 packets with Extension Headers?

- The data presented here suggests that EH drop could be the more significant issue here
- Perhaps we might want to think about advice to host stacks and applications to avoid EH altogether on the big-I Internet!

Thanks!